Synchronization and Control of Quantitative Systems

Laursen, Simon

DOI (link to publication from Publisher):
10.5278/vbn.phd.engsci.00182

Publication date:
2016

Document Version
Publisher's PDF, also known as Version of record

SYNCHRONIZATION AND CONTROL OF QUANTITATIVE SYSTEMS

BY
SIMON BORGBJERG LAURSEN

DISSERTATION SUBMITTED 2016

AALBORG UNIVERSITY
DENMARK

Synchronization and Control of Quantitative Systems

PhD Dissertation

Simon Borgbjerg Laursen

Aalborg University
Department of Computer Science
Selma Lagerløfs Vej 300
DK-9220 Aalborg

Dissertation submitted: September, 2016

PhD supervisor: Professor Kim Guldstrand Larsen
Aalborg University

Assistant PhD supervisor: Associate Professor Jirí Srba
Aalborg University

PhD committee: Associate Professor Manfred Jaeger (chairman)
Aalborg University

Associate Professor Thomas Troels Hildebrandt
ITU University of Copenhagen

Professor Jean-Francois Raskin
Université Libre de Bruxelles

PhD Series: Faculty of Engineering and Science, Aalborg University

ISSN (online): 2246-1248
ISBN (online): 978-87-7112-801-7

Published by:
Aalborg University Press
Skjernvej 4A, 2nd floor
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2016

Hofstadter's Law: *It always takes longer than you expect, even when you take into account Hofstadter's Law.*

– Douglas Hofstadter [76]

~

Abstract

Formal methods, such as model checking, theorem proving and static analysis, are used to provide confidence in system models and designs, as it can assist in all phases of system development and testing. However, formal modeling of software is not trivial, and the ever increasing demand for software functionality and the complexity of such software systems makes modeling even harder. This challenge creates the need for expressive and computationally feasible modeling and specification formalisms. In this thesis, we explore modeling formalisms for quantitative systems and systems with partial observability to determine their computational limitations and feasibility.

Initially, we study the synchronization problem, where the objective is to find a sequence of inputs that reset a system without knowing what state the system started in. This objective was first studied for systems where no information is available about its current system state. We extend this to systems with partial observability, where the controller has some information about the current state. The objective is to find a synchronizing strategy that, based on the observations, gives the next input. For deterministic systems, we show that the computational complexity of finding synchronizing strategies remains the same as for the classical problem with no observability. Furthermore, we show that for nondeterministic systems, the complexity increases as it is possible to encode alternation. We then extend the concept of synchronizing strategies to quantitative systems. In particular, we analyse when the controller has partial information about the current accumulated weight in a deterministic system. We prove the surprising result, that the existence of synchronizing strategy for such a system is decidable in polynomial time.

We also study quantitative games, i.e. games where part of the objective is to minimise or contratin some quantitative value. In particular average-energy games where objective aims to optimise the long-run average of the accumulated energy. We show that this objective arises naturally in several applications, and previous case studies. We prove that determining the winner in such games is in the intersection of NP and coNP and at least as hard as solving mean-payoff games. Then, we analyse cases where the system has to minimise the average energy while maintaining the accumulated energy within given bounds. Furthermore, we study average-energy games where the bounds are existentially quantified. Here, we consider the problem of determining upper bounds on the average accumulated energy and the capacity while satisfying a given bound. We show that the existence of a sufficient bound on the long-run average accumulated energy can be determined in doubly-exponential time. Lastly, we consider recharge games, a version of energy games, where all weights are negative and we have special recharge edges. For these games we show that the problem of bounding the long-run average energy is complete for exponential time, whereas the existential version of the problem is solvable in polynomial time.

Resumé

Formelle metoder, såsom model tjekking, bevisførelse og statisk analyse, bruges til at skabe tillid til systemmodeller og designs, og som værktøjer i alle faser af systemudvikling og test. Men formel modellering af software er ikke trivielt, og den stadigt stigende kompleksitet samt øget efterspørgsel af funktionalitet gør softwaremodellering endnu sværere. Denne udvikling skaber behov for udtryksfulde modellerings- og specifikationformalismer der samtidig er beregningsmæssigt mulige. I denne afhandling udforsker vi modelleringsformalismer for kvantitative systemer samt systemer med delvis observerbarhed. Derudover bestemmer vi modellers beregningsmæssige begrænsninger og opnåelighed.

Først studerer vi synkroniseringsproblemt, hvor målet er at finde en sekvens af input, der nulstiller et system uden at vide, hvilken tilstand systemet startede i. Dette problem er tidlgere blevet undersøgt for systemer, hvor der ikke er information om systemets aktuelle tilstand. Vi udvider dette til systemer med delvis observerbarhed, hvor controlleren har nogen viden om den aktuelle tilstand. Målet er nu at finde en synkroniserende strategi, der, baseret på observationer, giver det næste input. For deterministiske systemer, viser vi, at kompleksiteten af at finde en synkroniseringsstrategi forbliver den samme, som for det klassiske problem uden nogen observerbarhed. Endvidere viser vi, at for ikke-deterministiske systemer, stiger kompleksiteten, da det er muligt at indkode alternering. Derefter udvider vi begrebet synkroniseringsstrategier til kvantitative systemer. Specifikt analyserer vi når controlleren har delvis viden om den aktuelle akkumulerede værdi i et deterministisk system. Vi beviser overraskende, at eksistensen af en synkroninserende strategi er afgørtbart i polynomiel tid.

Vi studerer også kvantitative spil, nærmere præcist gennemsnitsenergispil. Målet i et gennemsnitsenergispil er at optimere det langsigtede gennemsnit af den akkumulerede energi. Vi viser, at dette opstår naturligt samt har en række tidligere anvendelser. Vi beviser, at vinderen af sådanne spil kan afgøres i fællesmængden mellem NP og coNP og at problemet mindst så svært som middelværdisspil. Derefter analyserer vi det tilfælde, hvor man ønsker at minimere den gennemsnitlige energi samt holde det akkumulerede energi niveau inden for nogle givende grænser. Vi betraget desuden gennemsnitsenergispil, hvor grænserne er eksistentielt kvantificerede. Her er problemet at bestemme en øvre grænser på den gennemsnitlige akkumulerede energi, hvor grænserne for energi niveauet er givet. Vi beviser, at det kan afgøres hvorvidt en sådan øvre grænse eksistere i dobbelt-eksponentiel tid. Sidst studere vi genopladningsspil, en version af energispil, hvor alle kanter enden har ikke positive vægte eller er særlige genopladningskanter. Vi viser at kombinationen af disse spil og gennemsnitsenergispil er fuldstændig for eksponentiel tid og i den eksistentielle udgave kan det afgøres i polynomiel tid.

Acknowledgments

First, and foremost I would like to thank my supervisors Kim Guldstrand Larsen and Jiří Srba. Thank you for taking me under your wings and making this PhD possible and for alle the guidance I have received during my studies, both as a Master and PhD student.

Next, I would like to send a big thanks to Patricia Bouyer and Nicolas Markey for making my four-month visit in Laboratoire Spécification et Vérification (LSV) at ENS Cachan pleasant, productive and inspiring. Furthermore, I would like to thank Mickael Randour for the valuable input and fruitful cooperation we had during my stay. I would also like to thank my co-authors Jan Křetínský and Martin Zimmermann. The work in this thesis would have been impossible without you.

Another big thanks goes to my fellow PhD student and long time office mate Erik Ramsgaard Wognsen for the talks about life as a PhD and for drinking rum with me on Fridays. The same thanks go to the fellow members of the "lunch club" and for sharing the lunch with me every day.

To my PhD friends Petr Novotný, Guillermo A. Pérez, Jan Krčál and Mahsa Shirmohammadi for always wanting to discuss, have driks and talk when we meet on PhD schools, conferences and workshops. To my friends in "Laks og Nødder" thank you for giving me a place where I do not have to think about tomorrow or the next paper deadline.

To my mom and dad, thank you for the endless support and help you have given me. Thanks to my little brother Kasper, for always being ready for a discussion and give insight into the area of computer science that we both find so interesting.

Finally, to my dear wife Cæcilie, thanks for bearing over with me in the hard times, for always loving me and supporting my choices. I am in no doubt, that without you, I would not have been able to complete my PhD. But most of all thank you for giving me the greatest gift of all, our wonderful daughter Thilde, who is the artists of the drawing on the front page.

Simon Borgbjerg Laursen
Aalborg University, September 2016

Thesis Details

Thesis Title:	Synchronization and Control of Quantitative Systems
PhD Student:	Simon Borgbjerg Laursen
Supervisors:	Professor Kim Guldstrand Larsen and Associate Professor Jiří Srba, Aalborg University

The main body of this thesis consists of the following papers.

[A] Simon Laursen, Kim Guldstrand Larsen, Jiří Srba, "Synchronizing Strategies under Partial Observability," at *25th International Conference on Concurrency Theory, (CONCUR 2014)*, LNCS vol. 8704, pp. 188–202, Springer, 2014.

[B] Simon Laursen, Jan Křetínský, Kim Guldstrand Larsen, Jiří Srba, "Polynomial Time Decidability of Weighted Synchronization under Partial Observability," at *26th International Conference on Concurrency Theory, (CONCUR 2015)*, LIPIcs vol. 42, pp. 142–154, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[C] Simon Laursen, Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim Guldstrand Larsen, "Average-energy games" in *Proceedings Sixth International Symposium on Games, Automata, Logics and Formal Verification, (GandALF 2015)*, EPTCS vol. 193, pp. 1–15, 2015. The full version in *Acta Informatica*, pp. 1–37, July 2016.

[D] Simon Laursen, Kim Guldstrand Larsen, Martin Zimmermann, "Limit Your Consumption! Finding Bounds in Average-energy Games," at *24th International Workshop on Quantitative Aspects of Programming Languages and Systems (QAPL 2016)*, EPTCS, 2016.

In addition to the main papers, the following publications have also been made.

- Simon Laursen, Kim Guldstrand Larsen, Jiří Srba, "Action Investment Energy Games," at *Mathematical and Engineering Methods in Computer Science, 8th International Doctoral Workshop, (MEMICS 2012)*, LNCS vol. 7721, pp. 155–167, Springer, 2012.

This thesis has been submitted for assessment in partial fulfillment of the PhD degree. The thesis is based on the submitted or published scientific papers which are listed above. Parts of the papers are used directly or indirectly in the extended summary of the thesis. As part of the assessment, co-author statements have been made available to the assessment committee and are also available at the Faculty. The thesis is not in its present form acceptable for open publication but only in limited and closed circulation as copyright may not be ensured.

Contents

Chapter 1

Introduction

Software systems are everywhere. They control homes, drive on roads and fly in the sky above us. We rely on software systems to do almost everything from heating the house to guiding us from A to B. The common thread in these computing systems is that they interact with the world around them. We group these systems under the common term *embedded system*. Embedded systems can be defined as engineering artifacts involving computation that are subject to physical constraints [73]. This definition ranges over all software based control systems that we all use every day, such as car brakes, heating systems, vacuum cleaners, pacemakers, elevators, etc.

There is evidence that we increasingly depend on embedded systems in our society, and it is clear that errors in these systems can have a costly outcome, cause injury or even lead to death. It is therefore of the utmost importance that reliability and correctness is ensured in these systems. Ensuring error-free and correct software is a non-trivial task; it is time-consuming and expensive to do. This is the case for all complex software systems including embedded systems. Correcting an error in thousands of deployed systems can be extremely costly if not impossible.

Testing is the common way for ensuring the quality and validating correctness of systems [81]. A common task for an embedded system is to monitor and respond to changes in the surrounding environment. A requirement of the system could be: "Whenever A is present do B". When doing testing, several test scenarios would be set up and executed. If all test scenarios resolve as expected, one would say that the system behaves correctly according to the tests, or in other words, the tests could not find any errors. The drawback of this approach is that completeness can not be ensured when testing, there may always be scenarios not covered by the test.

In contrast, the goal of *formal methods* is to verify and give an absolute insurance that a system contains no errors and only behaves as intended. This guarantee is given by a mathematically based method and is a much stronger result than the one from testing. However, formal methods are by their very nature performed on a model of the system, rather than on the actual system. The result, when using formal methods, is, therefore, only as good as the model of the system. This creates the need for expressive and computationally feasible modeling and specification formalisms, which is the main topic of this thesis.

The process of performing formal methods on a model is called model-based verification. We present an introduction to this area in the rest of this chapter.

1.1 Model Checking

Verification techniques for software were first studied in the 1960-70s. Most of these first attempts consisted of proof systems for programs, for instance, Hoare logic [74]. A program in this setting is seen as a function that, transforms an initial state into a final states in a sequence of steps. The goal of these proof systems was to derive guarantees for a program. This is done by giving assumptions on the input to the program and then, in a systematic manner, deriving guarantees about the output. A major problem with this lies in the assumption that a program is a finite sequence of steps that finishes and returns an output; this does not cover concurrency, where execution might be interleaved [104], or reactive systems where programs run forever interacting with it surroundings, which is the case for embedded system.

These limitations led to several developments, such as process algebra in the 70 and 80's [75, 98]. This made it possible to describe concurrent, reactive and distributed systems. Model checking as a verification technique for these systems was developed by Clarke and Emerson [57, 45], and Queille and Sifakis [109]. Contrary to the earlier attempts and other verification techniques from that time, model checking was fully automatic way to analysis finite state models. The initial goal for model checking was to prove the correctness of systems, the problem then was to define system correctness. This lead to the development of rich temporal specification formalisms such as computation tree logic CTL [46] and LTL [119]. Today, model checking and similar verification techniques are used to provide confidence in system models and designs, and it can assist in all phases of system development and testing. Model checking has successfully been used in the development of a wide range of systems, for instance, safety-critical medical systems [5], commercial software [32], network protocols [72], and hardware circuits [63]. The Turing Award in 2007 was given to Clarke, Emerson and Sifakis for their work on model checking. There are two comprehensive books on the subject of model checking, one by Clarke, Grumberg and Peled [47] the other by Baier and Katoen [6].

Essentially, model checking deals with two artifacts: a system model and a specification. A model describes the behavior of a system, and the specification describes the intended behavior of the system. Given a model of a system and a specification, model checking is then the process of proving or disproving if the model satisfies the specification. A model is given in a mathematical framework, often in the form of states and transitions [83]. A state in the model represents a state in the real system, and a transition describes how the model progresses from one state to another. The behavior of a model is then given by sequences of states. The set of all reachable states from a given initial state defines the state space. A specification is usually given as a formula in modal or temporal logic. A basic model checking algorithm would make an exhaustive search through the state space to find a sequence that violates the sequence described in the specification. If no such sequence is found the model checker returns that the system satisfies the specification.

1.2 Control Synthesis and Game Theory

Embedded systems often observe and interact with the environment surrounding them. House heating systems react when the temperature drops, and elevators move when they are requested on a different floor. Embedded systems can, therefore, be seen as control systems performing actions to meet some specification. Model checking can be used to verify that a given controller for a given environment satisfies its specification. The problem lies in creating such a controller; this is non-trivial for large systems and time-consuming. It is, therefore, desirable to automatically create the controller from the specification in such a way that the controller always satisfies the specification no matter how the environment behaves. This problem is considerably harder than model checking, as one has to construct a correct controller instead of checking if a proposed controller is correct. This process of automatically creating controllers or systems from the specification is called *synthesis*. The synthesis problems were first proposed and studied by Church for circuits [44] and later by Ramadge and Wonham [110] and Pnueli and Rosner [107] for reactive systems and control synthesis.

The synthesis problem can be formulated as a two player game where the *controller* plays against the *environment*. The specification is encoded into the *winning objective* for the controller, such that finding the *winning strategy* for the controller in the game is then equivalent to answering to the control synthesis problem. These games are played on graphs where the vertices are split into those owned by the controller and those owned by the environment. Each vertex represents a state of the real systems. The game starts by adding a pebble to the initial vertex and the player owning the vertex then moves the pebble to a successor vertex. This continues and the movement of the pebble creates a play that represents the behavior of the real systems. The synthesis question is then to decide if the controller has a strategy for choosing successors such that no matter how the environment behaves, the controller ensures that the play is winning. This game-theoretic formulation has become a standard tool for the synthesis of provable correct controllers for reactive systems and embedded systems.

Game theory in general provides the theoretical background for modeling interaction among agents. The framework of game theory has seen application in numerous disciplines as widespread as physiology, economics, biology, operations research and computer science. Morgenstern and von Neumann are seen as the pioneers and farthers of modern game theory with their book in 1944 [123], even though work by Cournot [48], Zermelo [124] and Borel [16] precedes them. Today, the subject of algorithmic game theory covers decision making, voting, auctioning, cooperation and other aspects, where multiple agents interact as teams or as opponents. A comprehensive book on the subject was published in 2007 by Nisan, Roughgarden, and Tardos [101].

For synthesis, the focus is on two-player zero-sum games played on graphs. Zero-sum defines the category of games where players are pure adversaries, meaning that if one player wins the other loses. Winning a game is defined by the winning objective, that describes the set of winning plays for one of the players. Winning objectives can be simple objectives such as reachability or complex such that Buchi or Parity. These games are grouped under the term ω-regular games. A comprehensive guide to current research was published by Grädel, Thomas, and Wilke [70].

1.3 Quantitative Systems

Embedded systems are by definition constrained by the physical platform and the environment. It is, therefore, essential when modeling such a system, that the modeling formalism used can cope with these constraints. Quantitative games [96, 12, 31, 111] are a common way to model resource constraints and performance requirements. The quantities represent measurable entities such as power consumption, response time or buffer size. In this setting, games are played on weighted graphs, where the winning objective is a map from plays to the numerical domain. This map can be seen as a payoff function, and the players have to either maximize or minimize the payoff depending on the objective. The classic objective to model consumption or usages is mean-payoff [26, 121, 126], where the objective is to minimize (or maximize) the mean of the encountered weights when a playing. Resource restrictions can be modeled using energy games [20, 59, 78], where the objective is to keep the accumulated weight within given bounds.

The conjunction between qualitative and quantitative objectives has also been studied for instance in mean-payoff parity [34], and energy parity [38] and a multi-dimensional combination of these [43]. In these games, the controller has two objectives; a quantitative mean-payoff or energy objective, conjoined with the qualitative objective of parity where the system must visit some specific states an infinite number of time. Other models, such as consumption systems [23] or battery transition systems [15], also lie in this category of dual objective games.

1.4 Imperfect Information

Before any system can interact with its environment in a meaningful way, it has to obtain information about it surroundings. For embedded systems, obtaining this information is often obtained by reading output from sensors or similar observers of the environment. Sensors always have some imprecision and in most cases, is it impossible or very expensive to obtain perfect information about the environment. Embedded systems therefore have to make decisions based on imperfect or partial information about the current state of the environment.

It is therefore necessary when modeling embedded systems, that the modeling formalism can cope with the concepts of imperfect information. This has been extensively studied for several different kind of models and assumptions of what is observable, for instance in the case of omega regular games [40], real-time systems [28] and probabilistic systems [102].

Another key feature of a system with limited or no observability is the ability to regain knowledge of its current state or return to a known state. For example, assume that an engineer knows how the system is supposed to operate, but does not know its current state. Such a scenario can occur when performing blackbox tests of an embedded system where the system state can not be observed. In this case, the engineer would benefit from having a controlled way to either reset the system to a known state, or give the system a sequence of inputs that reveals its state. The latter is the concept of homing sequences and has been widely studied [99, 67, 117]. The scenario of resetting is called synchronization [30, 122]. The synchronization problem is: Given a system deciding if it has a sequence of inputs (synchronizing word) that will bring it to a given state no matter in that state the systems was initially. The synchronizing problem was first studied by Ĉerný in 1964 and his conjecture regarding the length of synchronizing words is one of the longest standing open problems in automata theory. Problems in the area of synchronizing words [116, 97, 10], and in particular the study of synchronizing quantitative models, have seen a lot of attention lately [51, 50, 64].

1.5 Thesis Structure

The rest of this thesis is structured as follows:

Chapter 2: Synchronization Gives a formal introduction to the area of synchronization. We start by defining synchronizing words for finite automata and recall some classical problems and results within the area. We then introduce synchronizing of systems with partial observability, and present the results of the first two papers included in this thesis *Paper A: Synchronizing Strategies under Partial Observability* and *Paper B: Polynomial Time Decidability of Weighted Synchronization under Partial Observability.*

Chapter 3: Energy Games Introduces the concept of games on graphs and we define the concept of Energy Game and extension of Average Energy Games. We give an overview of the classical Energy Games problems, and present the new results of the two papers on Average Energy Games included in this thesis, *Paper C: Average-energy games* and *Paper D: Limit Your Consumption! Finding Bounds in Average-energy Games.*

Papers The second part of this thesis includes the full versions of all four papers.

Chapter 2

Synchronization

The concept of synchronizing words for finite automata has received lots of attention during the last years [116, 122, 103, 97]. A synchronizing word will, when read by an automaton, change the state of the automaton to a specific known state no matter where the automaton started. The theory has seen applications in testing [27], bio-computing [9], and robotics [3].

In this chapter we introduce the decision problems of synchronizing words and give an overview of the complexity results in the area. The following sections outline the results obtained in two first papers inclosed in this thesis, where we study synchronizing words under partial observability [95] and in a weighted setting [89]. We start by defining the model of nondeterministic finite automata.

Definition 2.1 (Nondeterministic Finite Automata)**.** *A* nondeterministic finite automaton *(NFA) is a triple* $\mathcal{A} = (Q, \Sigma, \delta)$ *where*

- Q *is a finite set of states,*
- Σ *is a finite input alphabet, and*
- $\delta \colon Q \times \Sigma \to 2^Q$ *is the transition function.*

The transition function is deterministic if $|\delta(s, a)| = 1$ for all $s \in Q$ and $a \in \Sigma$ in witch case we call it a deterministic finite automata (DFA). Similar, if it is partial $|\delta(s, a)| \leq 1$ for all $s \in Q$ and $a \in \Sigma$ then we call it a partial finite automata (PFA).

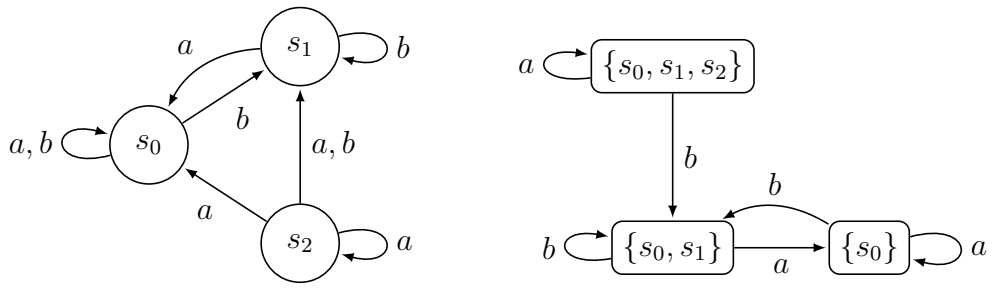
We extend the transition function to sets of states $S \subseteq Q$ such that $\delta(S, a) = \{\delta(s, a) \mid s \in S\}$ and by simple induction to range over sequences of input letters such that $\delta(S, \varepsilon) = S$ and $\delta(S, aw) = \delta(\delta(S, a), w)$, where $a \in \Sigma$ and $w \in \Sigma^*$.

We can now write $\delta(Q, w)$ and get the set of states we reach when starting from any state in the NFA and applying the word w. We then define the concept of a synchronizing word, as a word which brings the automaton from every state to the exactly same state.

Definition 2.2 (Synchronizing word)**.** *Given an NFA* $\mathcal{A} = (Q, \Sigma, \delta)$*, a word* $w \in \Sigma^*$ *is synchronizing for* $\mathcal{A}$ *if* $|\delta(Q, w)| = 1$*.*

The decision problem follows naturally:

Decision problem. *The **synchronization** problem, given an NFA (or PFA or DFA), decide if a synchronization word exists.*



(a) An NFA over the alphabet $\Sigma = \{a, b\}$

(b) The subset (knowledge) construction

Figure 2.1: An NFA and the subset (knowlege) construction over the NFA.

The synchronization problem for NFAs can be solved straightforwardly using the subset construction and searching for a path from the initial configuration to a configuration with only a singleton state inside. This construction is shown on Figure 2.1b for the NFA in Figure 2.1a. The subset construction provides a PSPACE membership, as it can be constructed and explored on-the-fly using only polynomial space. The lower bound of the problem can be shown via reduction from the PSPACE-complete problem of language intersection for finite automata [88]. The PSPACE-completeness result even holds for NFAs with only one nondeterministic transition and, even more surprising, also for PFAs. For DFAs the problem is NLOGSPACE-complete where its inclusion can be shown using the technique of pair-wise synchronization. This technique is a result of a lemma in Černý's first paper on synchronization [30]. The lemma says that iff any pair of states in the DFA can be synchronized then, all states can be. Algorithm 2.1 shows this procedure, it finds a synchronizing word for a DFA. If the synchronizing word is stored and returned then the algorithm uses polynomial time, otherwise it can be done in NLOGSPACE.

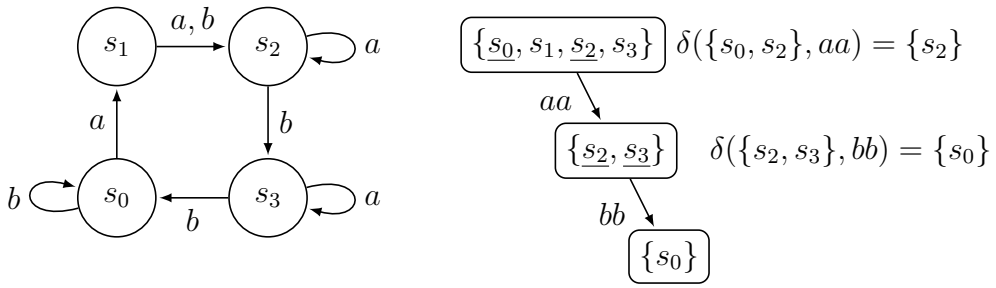
The algorithm is guaranteed to terminate, as the set S gets at least one state smaller in each iteration of the loop, given a runtime of at most $|Q|$ iterations. In step 5, the pair-wise synchronizing word is computed and at most $|Q|^2$ options are explored. The process is shown on Figure 2.2. The algorithm finds a synchronizing word if there is one, but not always the shortest one. An overview of the complexity results can be seen in Tabel 2.1.

Černý conjectured in 1964 that the length of the shortest synchronizing word for any n-state DFA is at most $(n-1)^2$ [30]. This conjecture is one of the longest standing conjectures in automata theory and despite numerous attempts in the last 50 years

Algorithm 2.1 Synchronization of DFAs.

Input: DFA $\mathcal{A} = (Q, \Sigma, \delta)$
Output: A synchronizing word w for $\mathcal{A}$, or FALSE

```
w ← empty sting
S ← Q
while S is non-singleton do
    Pick arbitrary s1, s2 ∈ S                          ▷ s1 ≠ s2
    Compute w' such that δ({s1, s2}, w' ∈ Σ) = {s}     ▷ Exploring Q × Q
    if no such w' exists return FALSE
    S ← δ(S, w')
    w ← ww'
return w
```



(a) An DFA over the alphabet $\Sigma = \{a, b\}$.

(b) Pair-wise synchronization given the word $aabb$.

Figure 2.2: A DFA and a possible synchronizing word, found using the pair wise synchronization. In each configuration are two states picked, these are underlined, and the word used to synchronize the two states is shown on the edge to the next configuration.

Complexity of Synchronization Problem

Type	Complexity
DFA	NLOGSPACE-complete [30, 122]
PFA	PSPACE-complete [97]
NFA	PSPACE-complete [116, 97]

Table 2.1: Complexity overview the Synchronization Problem.

no one has proved or disproved this. The best known upper bound by Pin [106] is $\frac{1}{6}(n^3 - n - 6)$. From the example DFA on Figure 2.2a there are several synchronizing words of different length for instance bbb or $aabb$. The decision problem related to the length of the synchronizing word is given below.

Decision problem. *The **short-synchronization** problem: Given an NFA (or PFA or DFA) and a bound $k \in \mathbb{N}$, decide if there exists a synchronizing word w such that $|w| \leq k$.*

The Short-Synchronization problem is NP-complete for DFAs, the hardness is proved by a simple reduction from 3-SAT [58], and the inclusion is by nondeterministically guessing the synchronizing word of length $\frac{1}{6}(n^3 - n - 6)$ (the bound by Pin) [58]. For PFA's and NFA's, the problem is still PSPACE-complete [97]. A complexity overview can be seen in Tabel 2.2.

Complexity of Short Synchronization Problem

Type	Complexity
DFA	NP-complete [58]
PFA	PSPACE-complete [97]
NFA	PSPACE-complete [97]

Table 2.2: Complexity overview the Short Synchronization Problem.

An alternative version of the synchronization problem is the subset synchronization problem, where a subset of states is given and the objective is to find a word such that this subset of states are synchronized into a single state. The problem is formally defined as follows.

Decision problem. *The **subset synchronization** problem: Given an NFA (or PFA or DFA) a subset $S \subseteq Q$, decide if there is there a word w such that $|\delta(S, w)| = 1$.*

The Subset Synchronization problem is PSPACE-complete even for DFAs [115]. Again the PSPACE-hardness comes from finite automata language intersection, as it is possible to pick the states where the synchronization starts. A similar construction to the one used for the synchronization problem for PFA or NFA can be used. A complexity overview is given in Table 2.3.

2.1 Synchronization and Partial Observability

For synchronizing words, the assumption is that the system can not be observed. This is a rather pessimistic assumption as it is often the case that the controller has some direct or indirect knowledge about the current state of the system. This knowledge gives the controller partial observability of the current state of the system.

Complexity of Subset Synchronization Problem	
Type	Complexity
DFA	PSPACE-complete [115, 117]
PFA	PSPACE-complete [115, 117]
NFA	PSPACE-complete [115, 117]

Table 2.3: Complexity overview the Subset Synchronization Problem.

Examples of such systems are sensor networks, where they only know the state of there immediate neighbors, systems with limited communication bandwidth where it is not possible to communicate the full state to the controller or systems where some external monitor can observe the state changes in the system, but the internal state is unknown.

In order to capture these more realistic scenarios, we extend the theory of synchronizing words, presented above, to synchronizing strategies that enable us to study the synchronization problems under partial observability. We deal with this problem in the setting of finite-state automata, where each state has a single observation from a finite set of observations, formally a Moore automaton [6].

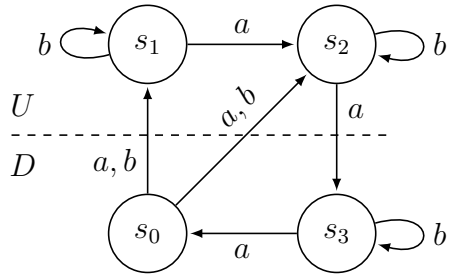
Definition 2.3 (Nondeterministic Moore automaton)**.** *A* nondeterministic Moore automaton *(NMA) is a tuple* $\mathcal{M} = (Q, \Sigma, \delta, \mathcal{O}, \gamma)$ *where*

- (Q, Σ, δ) *is an NFA,*
- $\mathcal{O}$ *is a nonempty observation set, and*
- $\gamma : S \to \mathcal{O}$ *is the observation function.*

Similar to the problem of synchronization words for NFAs, we distinguish between nondeterministic, partially deterministic and deterministic transition functions, given the three classes of Moore automata, NMA, PMA, DMA.

The objective is to find an adaptive strategy for generation of input letters based on the so-far seen observations. As for synchronizing words, the strategy should bring the system from any possible initial state to a single synchronizing state.

Formally, a strategy is a function from sequences of observations to input letters $\sigma : \mathcal{O}^+ \to \Sigma \cup \{done\}$ where $done \notin \Sigma$. The special output signal *done* is to show that the synchronizing state has been reached. The set of states reached when applying the strategy σ until the special signal *done* is reached is defined as $\sigma[S]$, where $S \in Q$ is the subset of states started from. We can now define what a synchronizing strategy is.



(a) An NMA with no synchronizing word.

Synchronizing strategy σ		
$\sigma(D)$	$\to$	b
$\sigma(DD)$	$\to$	*done*
$\sigma(DU)$	$\to$	a
$\sigma(DUU)$	$\to$	a
$\sigma(DUD)$	$\to$	*done*
$\sigma(DUUD)$	$\to$	*done*
$\sigma(U)$	$\to$	a
$\sigma(UU)$	$\to$	a
$\sigma(UD)$	$\to$	*done*
$\sigma(UUD)$	$\to$	*done*

(b) Synchronization strategy, given the observations U and D.

Figure 2.3: An example of a NMA and a synchronizing strategy σ that brings the system to the state s_3 no matter in witch state it initial started.

Definition 2.4 (Synchronizing strategy)**.** *Given an NMA $\mathcal{M} = (Q, \Sigma, \delta, \mathcal{O}, \gamma)$, a strategy σ is* synchronizing *if $\sigma[Q]$ is a singleton set.*

An example of NMA and its synchronizing strategy can be seen on Figure 2.3. The strategy brings the automaton on the state s_3. Note that the automaton does not have any synchronizing word, and that observations *up* (U) and *down* (D) are necessary in order for it to have a synchronizing strategy.

We can now define the decision problem for synchronization under partial observability.

Decision problem. *The **synchronization** problem under partial observability: Given an NMA (or PMA or DMA), decide if a synchronizing strategy exists.*

We prove in this thesis that the problem is EXPTIME-complete for NMAs. The containment comes by translating to a two-player knowledge game, where finding a memoryless winning strategy in this knowledge games gives the synchronizing strategy for our original problem. This knowledge game is played on an arena constructed similar to the subset construction used for finding synchronizing words on NFAs. The games comes in as the opponent resolves the nondeterminism. We prove the hardness by a reduction from the acceptance problem of alternating linear bounded automata.

For DMA and PMA, we introduce the concept of an aggregated knowledge graph and use it to derive a PSPACE containment for PMA and NLOGSPACE containment for DMA, despite the double-exponential size of the aggregated knowledge graph. These results match the lower bounds for synchronizing words, making the bounds complete. An overview of our results can be found in Table 2.4.

Synchronization under Partial Observability	
Type	Complexity
NMA	NLOGSPACE-complete
PMA	PSPACE-complete
NMA	EXPTIME-complete

Table 2.4: Complexity result of Synchronization under Partial Observability for different types of Moore automata.

The second problem of short-synchronization moreover asks about the existence of a strategy shorter than a given length bound. The length of the strategy σ, denoted $\mathit{length}(\sigma)$, is defined as the longest possible play before the action *done* is reached. The problem is defined as follows:

Decision problem. *The **short-synchronization** problem under partial observability, given an NMA (or PMA or DMA) and a bound $k \in \mathbb{N}$, decide if there is a synchronizing strategy σ such that $\mathit{length}(\sigma) \leq k$?*

We show that the complexity here follows the same patten as for the synchronization problem, the general problem for NMA is EXPTIME-complete, for PMA and DMA the problem stays in the same complexity as for synchronization word, namely PSPACE-complete and NP-complete. See Table 2.5 for an overview.

Short-Synchronization under Partial Observability	
Type	Complexity
NMA	NP-complete
PMA	PSPACE-complete
NMA	EXPTIME-complete

Table 2.5: Complexity result of Short-Synchronization under Partial Observability for different types of Moore automata.

Finally, the subset synchronization problem asks to synchronize only a subset of states to a single synchronizing state.

Decision problem. *The* ***subset-synchronization*** *problem under partial observability: Given NMA (or PMA or DMA) and subset $S \subseteq Q$, decide if there is a strategy σ for $\mathcal{M}$ such that $|last(\sigma[S])| = 1$?*

The subset-synchronization problem under partial observability is EXPTIME-complete for NMA, where it is PSPACE-complete for both PMA and DMA. See Table 2.6 for an overview.

Subset-Synchronization under Partial Observability	
Type	Complexity
NMA	PSPACE-complete
PMA	PSPACE-complete
NMA	EXPTIME-complete

Table 2.6: Complexity result of Subset-Synchronization under Partial Observability for different types of Moore automata.

The complexity lower-bounds for the classical word synchronization transfer to the more general setting with partial observability. We are able to match the lower-bounds with corresponding upper-bounds for the cases of DMA and PMA, where the transition relation is deterministic. In case of nondeterministic systems, the three synchronization problems become EXPTIME-complete.

2.2 Weighted Synchronization

Synchronization of weighted and quantitative systems has received attention recently, for instance in the case of stochastic and probabilistic automata [51, 50, 53, 52], weighted automata with positive weights [64], integer weighted and timed automata in [49], and in the case of weighted automata with matrix-labeled transitions [77].

In general, it is impossible of find a synchronizing word for deterministic weighted automata, where both the states and accumulated weight must be synchronized. For instance, assume that we are in the same location ℓ with weights deferring only by one, (ℓ, z) and $(\ell, z+1)$, any word will by the assumption of determinism maintain the relative difference in the weights, hence never synchronize them. To overcome this challenge, we therefore introduce the concept of partial observability of the accumulated weight. First, we formally define the model of weight automata and their semantics.

Definition 2.5 (Weighted Automaton)**.** *A (deterministic)* weighted automaton (WA) *is a tuple* $\mathcal{A} = (L, Act, E, W)$ *where*

- L *is a finite set of locations,*
- Act *is a finite set of actions,*
- $E : L \times Act \to L$ *is a transition function, and*
- $W : L \times Act \to \mathbb{Z}$ *is a weight function.*

A *state* of a weighted automaton is a pair $(\ell, z) \in L \times \mathbb{Z}$ where ℓ is the current location and z the current accumulated weight. We write $(\ell, z) \xrightarrow{a,w} (\ell', z')$ if $E(\ell, a) = \ell'$, $W(\ell, a) = w$ and $z' = z + w$.

The observation function $\gamma\colon L \times \mathbb{Z} \to \mathcal{O}$ is a map from states (location, weight) to an observation from the non-empty set $\mathcal{O}$. Note that if $\mathcal{O}$ only have a single element, it would not be possible to synchronize the system, as described above. We now limit us to the study where $\mathcal{O}$ is finite and we define a *strategy* similarly to the case without weights, $\sigma : \mathcal{O}^+ \to Act \cup \{done\}$ where $done \notin Act$, signaling that no further actions will be proposed. We similar define the set of states reached when it is applying the strategy σ until the special signal *done* as $\sigma[S]$, where $S \subseteq L$.

Definition 2.6 (Synchronizing strategy)**.** *Give a WA* $\mathcal{A} = (L, Act, E, W)$*, an observation function* γ *and the observation set* $\mathcal{O}$*, a strategy* σ *is* synchronizing *if* $\sigma[Q]$ *is a singleton set.*

We can now define the decision problem of weighted synchronizing under partial observability:

Decision problem. *The* ***weighted-synchronization*** *problem under partial observability: Given a WA* $\mathcal{A}$ *and an observation function* γ*, decide if there is a synchronizing strategy* σ *for* $\mathcal{WA}$*.*

To study the complexity of this decision problem, we focus on the simplest non-trivial observation function. We define this observation function γ as a map to the observation set $\mathcal{O} = \{<0, \geq 0\}$, such that

$$\gamma((\ell, z)) = \begin{cases} <0 & \text{if } z < 0 \\ \geq 0 & \text{if } z \geq 0 \ . \end{cases}$$

This minimal observation function returns whether the accumulated weight is positive or negative. We show that the synchronization problem for deterministic weighted automata under (minimal) partial observability is decidable in polynomial time. We do this by stating five necessary properties that the weighted automaton needs

to satisfy for it to have a synchronizing strategy. Moreover, we show that these properties are sufficient and that they all can be checked in polynomial time.

One of the properties requires the existence of reachable +1 and -1 cycles in the weighted automaton. The algorithm for finding +1 and −1 cycles relies on finding a positive and a negative cycle and furthermore deciding if the greatest common divisor of all cycles in a graph is 1. The greatest common divisor of all cycles of a graph is also know as the *period*.

Deciding presence of a positive and negative cycle and producing their witness can be done using a Bellman-Ford algorithm in polynomial time. Finding the period of a graph can also be computed in polynomial time. The result for unweighted graphs (all weights are one) was proven by Knuth [84]. An extension to weighted graphs was suggest but not proven in [4]. Furthermore Knuth's work is not widely accessible as the only hardcopy of the report is located at the library of Stanford University, with no electronic copy online. We therefore also provide our own proof, using different techniques.

2.3 Main Contributions

We now briefly summarize the main contributions of this thesis within the area of synchronization under partial observability.

First, we have introduced synchronization under partial observability and synchronizing strategies. We have analyzed the complexity the three synchronization problems in-depth. We have found lower-bounds matching the original problem of synchronization words and corresponding upper bounds for the cases of DFA and PFA, where the transition relation is deterministic. For the case of nondeterministic systems, we found that the studied synchronization problems moved to a higher complexity class, as the combination of nondeterminism with partial observability allows us to encode alternation.

Second, we have extended the work of synchronization under partial observability to the quantitative setting, where the systems are modeled by weighted automata. We showed that the synchronization problem for deterministic weighted automata under minimal partial observability is decidable in polynomial time, even though the resulting synchronization strategy may be unbounded as it depends on the initial weight value. The result is based on a polynomial time algorithm for deciding the existence of +1 and −1 cycles in a weighted graph.

Chapter 3

Energy Games

Quantitative games in general [12, 111, 25], and energy games, particular [31, 18, 78, 105], have seen increasing attention in the last decade and lately several extensions of games where they are combined with qualitative games have been studied. These multi-objective games create a basis for coping with and modelling resource bounded systems with multiple restrictions and tradeoffs between them.

In this chapter, we recall the concept of energy games and give an overview of the known complexity results to some extensions of energy games. Then, we focus on a new variant called average-energy games that is introduced in the thesis. The chapter is based on the papers on average-energy games inclosed in this thesis [21, 22, 94].

First, we formally define the concept of a two player game. A two player game is played on a graph area, where we name the players *Eva* and *Adam*. Formally an arena is defined as follows.

Definition 3.1 (Arena)**.** *A* game arena *is a tuple* $\mathsf{A} = (V, V_{Eva}, V_{Adam}, E, v_I)$ *where*

- (V, E) *is a finite directed graph,*
- $V = V_{Eva} \uplus V_{Adam}$ *is a disjoint partition of vertices and*
- $v_I \in V$ *is the initial vertex.*

The vertices in V_{Eva} belong to *Eva* and are drawn as circles, whereas vertices in V_{Adam} belong to *Adam* and are drawn as rectangles. A play in an arena A is an infinite path $\pi = v_0 v_1 v_2 \cdots$ with $v_0 = v_I$ and for all $i \geq 0$, $(v_i, v_{i+1}) \in E$. A *game* $\mathcal{G} = (\mathsf{A}, \mathrm{Win})$ consists of an arena A, and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for *Eva*, the *objective* of $\mathcal{G}$.

To determine who wins the game, we need to define the concept of a *strategy*. In general, a strategy is a function that, based on a play history, gives the next move. Formerly, a strategy for *Eva* is the map $\sigma_{Eva} \colon V^* V_{Eva} \to V$ such that $(v, \sigma_{Eva}(wv)) \in E$ for all $wv \in V^* V_{Eva}$, the strategy for *Adam* is defined similarly.

We say that a play $v_0 v_1 v_2 \cdots$ is *consistent* with a strategy σ_i for Player i if all successors are chosen by Player i in accordance with the strategy $v_{n+1} = \sigma_i(v_0 v_1 \cdots v_n)$ for every n with $v_n \in V_i$.

Eva is the winner of the game $\mathcal{G} = (\mathsf{A}, \mathrm{Win})$ if there exists a strategy σ_{Eva} such that for any strategy for *Adam* σ_{Adam}, the resulting play is consistent with σ_{Eva} and σ_{Adam} is in the winning set Win. We say that *Eva* wins $\mathcal{G}$ if she has a winning strategy for $\mathcal{G}$. We can now define the decision problem for at general game.

Decision problem. *The* ***general game*** *problem: Given a game* $\mathcal{G} = (\mathsf{A}, \mathrm{Win})$*, decide if there exists a winning strategy for Eva.*

In energy games, the objective of *Eva* is to make sure that the accumulated weight of a play is bounded by some constraints, where the objective for *Adam* is to reach a state where the accumulated weight breaks these constraints. The weights are added to the edges of the arena, through the introducing of a weight function. Given an arena $(V, V_{Eva}, V_{Adam}, E, v_I)$ a weight function $W \colon E \to \mathbb{Z}$ maps every edge to an integer weight. The weights are encoded in binary.

We can now, given a play prefix, calculate the current energy level by adding the weight along the play path together. Formally the energy level of a play prefix $v_0 \cdots v_n$ is the accumulated weight of its edges, i.e.,

$$\mathrm{EL}(v_0 \cdots v_n) = \sum_{i=0}^{n-1} W(v_i, v_{i+1})$$

Using the energy level, we can now define the two classic energy objectives namely, (1) lower bounded and (2) lower- and upper-bounded energy. The simplest is the lower-bounded energy objective that requires *Eva* to keep the energy level non-negative:

$$\mathsf{Energy}_{\mathsf{L}}(W) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \forall n.\, 0 \leq \mathrm{EL}(v_0 \cdots v_n)\}.$$

The lower- and upper-bounded energy objective requires *Eva* to always keep the energy level between 0 and some given upper bound $cap \in \mathbb{N}$:

$$\mathsf{Energy}_{\mathsf{LU}}(W, cap) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \forall n.\, 0 \leq \mathrm{EL}(v_0 \cdots v_n) \leq cap\}.$$

As seen in the game arena Figure 3.1. Here the game starts in vertex a with the accumulated energy of 0. Assume that the winning condition is a lower-bounded energy $\mathsf{Energy}_{\mathsf{L}}$ (we often omit the weight function as it is given from the context). In the vertex a *Eva* has two options; either to take the loop of -1 and resulting in the energy level to drop below 0 or take the $+3$ edge to b. It is evident that taking the loop will lose her the game, therefore the only winning option is to follow the $+3$ edge to b. It is clear that in b is it impossible for *Adam* to force *Eva* to lose, as the loop does not change the energy level, and the -2 edge changes the energy level to 1. In c, there are no other option for *Adam* than to return the play to a. Based on this example is the winning strategy for *Eva* is clear; she just always has

to take the +3 edge when in vertex a, and she is thereby making her the winner of this little game. A strategy of this type is called memoryless, as the decision of what to do next is only based on the current state of the game. lower-bound energy games are memoryless determined [31], meaning that it suffices to consider memoryless strategies to determine who wins the game. This is not the case for lower- and upper-bounded energy games as the next example will show.

Again, we use the arena in Figure 3.1. Now assume that the winning objective is lower- and upper-bounded energy $\mathsf{Energy}_{\mathsf{LU}}$, with the capacity bound $cap = 3$. If *Eva* follows the memoryless strategy from before, it is clear that she loses when she arrives in a the second time and takes the edge to b, since the energy level increases to 4 and breaks the capacity bound. To compensate for this, she has to bring the energy down to 0 before taking the +3 edge to b using the -1 loop on a. This gives her a winning strategy to the $\mathsf{Energy}_{\mathsf{LU}}$ game, and shows that memory is required for *Eva* to win the game, as she has to know the current energy level to make the correct decision in the vertex a. We prove in this thesis that pseudo-polynomial memory is both sufficient and necessary for *Eva* to win.

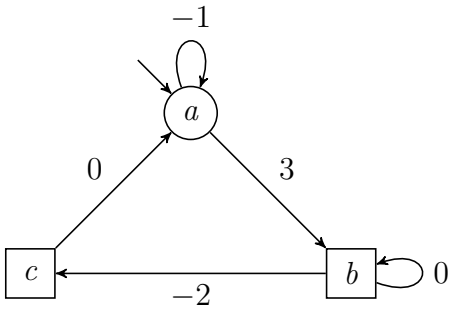


Figure 3.1: Simple example of game arena with weights on edges.

A complexity overview of the following two decision problem can be found in Table 3.1.

Decision problem. *The* ***lower-bounded energy game*** *problem: Given a game* $\mathcal{G} = (\mathsf{A}, \mathsf{Energy}_{\mathsf{L}}(W))$*, decide if Eva have a winning strategy* σ_{Eva}*?*

Decision problem. *The* ***lower- and upper-Bounded energy game****: Given a game* $\mathcal{G} = (\mathsf{A}, \mathsf{Energy}_{\mathsf{LU}}(W, cap))$ *decide if Eva have a winning strategy* σ_{Eva}*?*

Complexity of energy games		
Objective	**1-player**	**2-players**
$\mathsf{Energy}_{\mathsf{L}}$	PTIME [18]	NP ∩ coNP [31, 18]
$\mathsf{Energy}_{\mathsf{LU}}$	PSPACE-complete [61]	EXPTIME-complete [18]

Table 3.1: Complexity overview of energy games.

3.1 Average-energy Games

The two energy game objectives introduced above are safety objectives that ensure that nothing bad happens i.e. not running out of energy or exceeding some fixed capacity. The energy objective can be used to model such safety requirements. However, in most cases, it is desirable to make the system perform some task while being safe. This can, for instance, be done by mixing winning objectives. An example of this is energy parity games [38], where both a parity objective and an energy objective must be satisfied for *Eva* to win.

In this thesis, we introduce another quantitative winning objective, namely average-energy. This objective relates to the long-run average of the accumulated energy. This objective can be directly translated into the requirements in the Hydac case study from the *Quasimodo* project [29]. The case study focus on controlling an oil pump and thereby maintaining the oil level in the tank within some given safety limits and at the same time keeping the long term average level low as possible. An illustration of the tank can be seen on Figure 3.2. In the case study, the environment was given as a fixed schedule. It would, however, be possible to model much more sophisticated non-deterministic environments using average-energy games.

Formally, the average-energy objective requires *Eva* to keep the long-run average of the accumulated energy below a given threshold t:

$$\mathsf{AvgEnergy}(W, t) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathrm{EL}(v_0 \cdots v_i) \leq t\}$$

This objective is interesting by itself as it can be seen as a refinement of the total-payoff objective, in the same way as the total-payoff is seen as a refinement of the mean-payoff objective. This refinement is illustrated on Figure 3.3, where two plays with the same mean-payoff and total-payoff have different average-energy. We also show that average-energy games are memoryless determined.

We then conjoin the average-energy objectives with the lower bounded energy to create the lower bounded average objectives

$$\mathsf{AvgEnergy}_{\mathsf{L}}(W, t) = \mathsf{Energy}_{\mathsf{L}}(W) \cap \mathsf{AvgEnergy}(W, t).$$

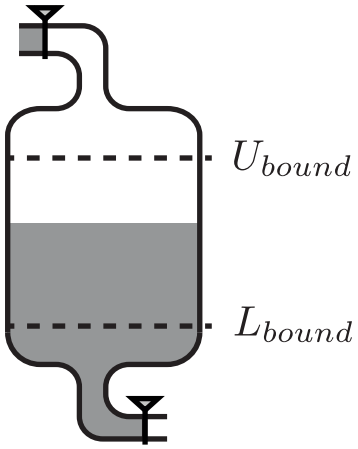
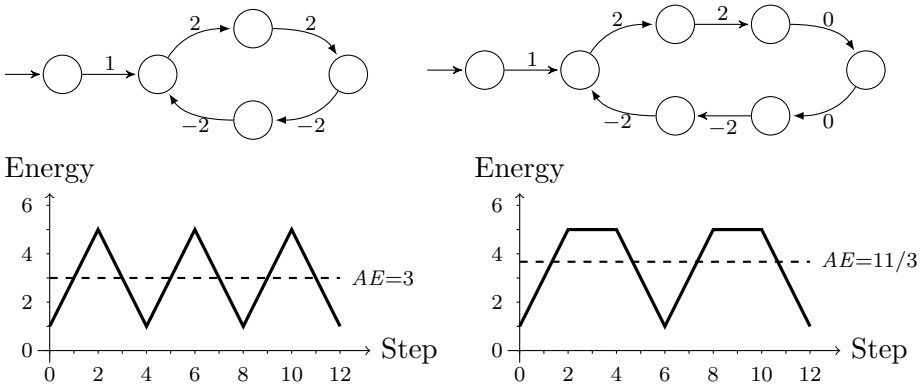


Figure 3.2: An illustration of the oil tank from the Hydac case study from the *Quasimodo*. The objective was to keep the oil level in the tank between the two bounds U_{bound} and L_{bound}, and at the same time keep the oil level as low as possible. It was possible to control the pump that supplied oil to the tank in the top, but the output from the tank was controlled by the environment.



(a) Play π_1 sees energy levels $(1, 3, 5, 3)^\omega$ giving an average-energy of 3.

(b) Play π_2 sees energy levels $(1, 3, 5, 5, 5, 3)^\omega$ giving an average-energy of 11/3.

Figure 3.3: Both plays have identical mean-payoff of 0 and total-payoff of 5, however the average-energy (AE) is different for the two plays.

Similar for the lower- and upper-bounded average-energy objective

$$\mathsf{AvgEnergy}_{\mathsf{LU}}(W, cap, t) = \mathsf{Energy}_{\mathsf{LU}}(W, cap) \cap \mathsf{AvgEnergy}(t).$$

The complexity overview of deciding the winner of these games is given in Table 3.2 below.

The complexity lower bound for the 2-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem still remain open.

Complexity of Average-energy Games

Objective	**1-player**	**2-players**
AvgEnergy	PTIME	NP ∩ coNP
$\mathsf{AvgEnergy}_{\mathsf{LU}}$, polynomial *cap*	PTIME	NP ∩ coNP
$\mathsf{AvgEnergy}_{\mathsf{LU}}$, arbitrary *cap*	PSPACE-complete	EXPTIME-complete
$\mathsf{AvgEnergy}_{\mathsf{L}}$	∈ PSPACE / NP-hard	*open* / EXPTIME-hard

Table 3.2: Complexity overview of average-energy games.

To get one step closer to solve this problem have we analyzed the existential version of this particular problem. This analysis lead to the finding in the following section.

3.2 Finding Upper-Bounds

We further study the average-energy games by analyzing an existential version of the decision problem, where the average threshold is existentially quantified. We analyze the problem of deciding whether there exists a threshold to which *Eva* can bound the long-run average accumulated energy while keeping the accumulated energy non-negative. Formally the problem is defined as follows:

Decision problem. *The* ***existential threshold problem*** *for lower-bounded average-energy game: Given an arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$ *and a weight function* $W \colon E \to \mathbb{Z}$*, decide if there exists a threshold* $t \in \mathbb{N}$ *such that Eva wins* $(\mathsf{A}, \mathsf{AvgEnergy}_{\mathsf{L}}(W, t))$*?*

Now observe the game arena on Figure 3.4. To solve the existential threshold problem we have to determine if there exists a threshold t such that *Eva* wins. We can in this simple example do this by observing the state b. From b is there a positive cycle of $+2$ when going to to c and back, and there is a zero cycle going to d and back. If d was not controlled by *Adam* this would solve our problem as the path $abc(bd)^\omega$ would bound the average-energy by 1.5. However, d is controlled by *Adam*, and *Eva* does not control *Adams* strategy there. To ensue that *Adam* does not take the -4 transition from d does *Eva* have to arrive there with an energy level on 4 or above. This can be done by taking the transition from c to d and then starting the zero loop, giving the path $abc(db)^\omega$ and an average-energy of 6.5. This can be improved if *Eva* instead takes the $+2$ cycle twice giving the path $abcbc(bd)^\omega$ with an average-energy of 3.5.

We prove that in general can this problem can be solved in doubly-exponential time. The proof is a polynomial time translation into a upper- and lower-bounded energy

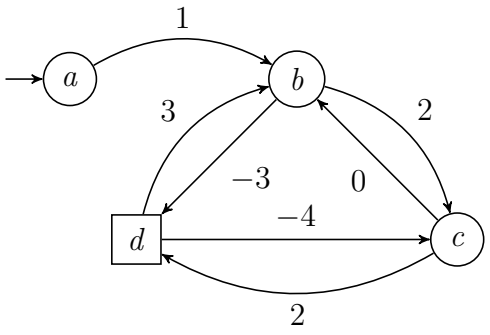


Figure 3.4: Another example of a game arena.

game with two components, i.e each edge vector of two weights, a problem which is known to be in 2EXPTIME.

3.2.1 Recharge Games

The model of recharge games is a variation of energy games where are all weights are non-positive, and there are special recharge edges that recharge the energy to some given capacity. Recharge games are similar to consumption games [24], but here *Eva* picks the new energy level while traversing a recharge edge. For recharge games we define the weight function gives a non-positive weight or a special recharge action R, i.e., $W \colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$. The recharge action R returns the energy level to a given upper bound capacity we name this *cap*.

For recharge games is the energy level defined as the energy left since the last recharge action. Formally, $\mathrm{EL}_{cap}(v_0 \cdots v_n) = cap + \mathrm{EL}(x)$, where x is the longest suffix of $v_0 \cdots v_n$ without an R-edge, i.e., $W(v_j, v_{j+1}) \neq \mathrm{R}$ for all (v_j, v_{j+1}) in x. All plays start which starts with energy level *cap*. We can now define the objective of a recharge game as

$$\mathsf{Recharge}(W, cap) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \forall n.\, \mathrm{EL}_{cap}(v_0 \cdots v_n) \geq 0\}.$$

We combine recharge games with the average-energy objective given the combined objective of average-bounded recharge games

$$\mathsf{AvgRecharge}(W, cap, t) =$$
$$\{v_0 v_1 v_2 \cdots \in V^\omega \mid \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathrm{EL}_{cap}(v_0 \cdots v_i) \leq t\} \cap \mathsf{Recharge}(W, cap).$$

We show that this combination becomes EXPTIME-hard to solve when the average threshold is a part of the input. To overcome the high complexity, we consider the problem where the recharge capacity is also existentially quantified. We show that

then problem then is solvable in polynomial time by a reduction to a three-color parity games.

3.3 Main Contributions

We now briefly summarize the main contributions of this thesis within the area of average-energy games.

First, we have presented a thorough study of the average-energy objective and showed that average-energy games belong to the same complexity class as mean-payoff, total-payoff and lower-bounded energy games. Furthermore, we have studied average-energy games with lower- and upper-bounded energy and provided preliminary results for the case of average-energy with a lower bound. We also provided a study of average-energy games where the bound on the average is existentially quantified instead of given as part of the input.

Second, we showed that this problem is equivalent to determining whether the maximal energy level can be uniformly bounded by a strategy, a problem known to be solvable in doubly-exponential time.

Finally, we considered a different type of energy evolution, where energy is only consumed or reset to some fixed capacity. We showed that solving the average-bounded variants of these games is complete for the exponential time. We also proved that the variant where the upper bound is existentially quantified can be solvable in polynomial time. In addition, we studied tradeoffs between the different bounds and the memory requirements of winning strategies.

Papers

Paper A

Synchronizing Strategies under Partial Observability

Simon Laursen Kim G. Larsen Jiří Srba
Aalborg University, Department of Computer Science, Denmark

Abstract Embedded devices usually share only partial information about their current configurations as the communication bandwidth can be restricted. Despite this, we may wish to bring a failed device into a given predetermined configuration. This problem, also known as resetting or synchronizing words, has been intensively studied for systems that do not provide any information about their configurations. In order to capture more general scenarios, we extend the existing theory of synchronizing words to synchronizing strategies, and study the synchronization, short-synchronization and subset-to-subset synchronization problems under partial observability. We provide a comprehensive complexity analysis of these problems, concluding that for deterministic systems the complexity of the problems under partial observability remains the same as for the classical synchronization problems, whereas for nondeterministic systems the complexity increases already for systems with just two observations, as we can now encode alternation.

Publication History The paper was published in the Proceedings of the 25th International Conference on Concurrency Theory, (CONCUR 2014), in the Lecture Notes in Computer Science, LNCS vol. 8704, pp. 188–202 by Springer, 2014. This thesis includes a full version of the paper with all proves and a modified layout.

The layout has been revised.

A.1 Introduction

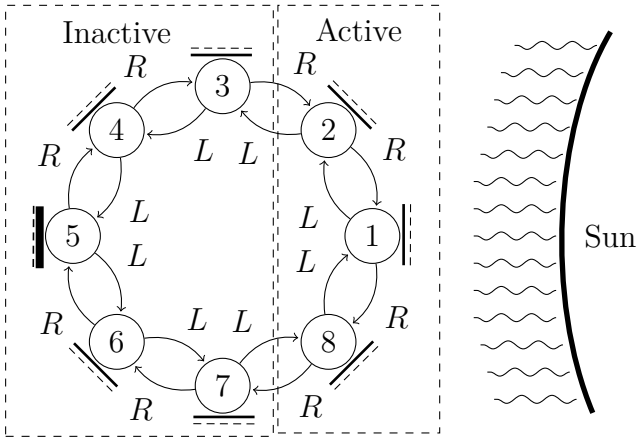
In February last year (2013), Aalborg University launched an experimental satellite [13] designed by students. There was a failure during the initialization phase executed by the satellite at the orbit, resulting in unknown orientation of the solar panel. This caused significant problems with energy supply and very limited communication capabilities of the satellite, especially when transmitting information that is energetically more expensive than receiving it. The task was to command the satellite from the Earth so that it returned to some predefined well-known position.

A simplified model of the problem is depicted in Figure A.1a. In the example, we assume for illustration purposes that there are only eight possible rotation positions of a single solar panel, numbered by 1 to 8 in the figure. The thin lines with a dashed surface indicate the direction the panel is facing in a given position. This determines whether the panel is active and produces energy (facing towards light) or inactive and does not produce any energy. The thick line at position 5 indicates the current (unknown) position of the solar panel. The satellite cannot communicate the exact position of the solar panel, instead it is only capable of transmitting information as to whether the current position produces energy (observation Active) or not (observation Inactive). The panel can be commanded to rotate one position to the left (action L) or to the right (action R) and our task is to bring it from any possible (unknown) position into the position 1 where it produces most energy. As we cannot observe the actual position of the panel, we need to find a strategy that relies only on the fact whether the panel is Active or Inactive. Such a strategy indeed exists as shown in Figure A.1b.

The classical concept of synchronizing words [30] for deterministic finite automata dates back more than 50 years and it concerns the existence of a word that brings a given automaton from any of its states to a single state (see [117, 122] for recent survey papers). However, for our example in Figure A.1a it is clear that there is no single synchronizing word—in this classical setting—over $\{L, R\}$ that can bring the panel into the same position. Instead, we need to design a strategy that relies on a partial information about the system, in our case on whether the panel is Active or Inactive.

A.1.1 Our Contribution

We introduce a general synthesis problem for synchronizing *strategies* of systems with *partial observability*. We deal with this problem in the setting of finite-



(a) Unknown orientation of the satellite's solar panel

Synchronizing strategy that brings the system to state 1:

if Inactive **then**
  repeat rotate L
  until Active;
else repeat rotate R
  until Inactive;
  rotate L;
endif
rotate L;

(b) Synch. strategy

Figure A.1: Satellite with partial observability and its synchronizing strategy.

state automata where each state has a single observation from a finite set of observations; we call the model *labelled transition system with partial observability* (LTSP). The task is to suggest a strategy for adaptive generation of actions based on the so-far seen observations. Such a strategy should bring the system from any possible initial state into a single synchronizing state. We also consider two other variants of the synchronization synthesis problem (i) with a bound on the maximal length of (runs of) the strategy (short-synchronization) and (ii) synchronization from a given set of states to a given set of states (subset-to-subset synchronization). We provide a comprehensive complexity study of these problems in the setting of total deterministic (DFA), partial deterministic (PFA) and nondeterministic (NFA) finite automata. Our results, compared to the classical synchronization problems, are summarized in the right column of Figure A.1.

Our first technical contribution is a translation from the synthesis of history-dependent synchronizing strategies on LTSP to the synthesis of memoryless winning reachability strategies for a larger two-player knowledge game. This allows us to argue for the EXPTIME containment of the synchronization problem on NFA. However, for DFA and PFA the knowledge game is insufficient to obtain tight complexity upper-bounds. For this reason, and as our second contribution, we define a notion of aggregated knowledge graph allowing us to derive a PSPACE containment for PFA and NL containment for DFA, despite the double-exponential size of the aggregated knowledge graph in the general nondeterministic case.

In order to complement the complexity upper-bounds with matching lower-

		Classical synchronization No information, $\lvert\mathcal{O}\rvert = 1$	**Partial Observability** No restriction on $\mathcal{O}$
Synchronization	DFA	NLOGSPACE-complete [30, 122]	**NLOGSPACE-complete** (Thm. A.13)
	PFA	PSPACE-complete [97]	**PSPACE-complete** (Thm. A.12)
	NFA	PSPACE-complete [116, 97]	**EXPTIME-complete** (Thm. A.7, A.15)
Short-synch.	DFA	NP-complete [58]	**NP-complete** (Thm. A.13)
	PFA	PSPACE-complete [97]	**PSPACE-complete** (Thm. A.12)
	NFA	PSPACE-complete [97]	**EXPTIME-complete** (Thm. A.7, A.15)
Subset-to-subset	DFA	PSPACE-complete [115]	**PSPACE-complete** (Thm. A.12)
	PFA	PSPACE-complete ([115])	**PSPACE-complete** (Thm. A.12)
	NFA	PSPACE-complete ([115])	**EXPTIME-complete** (Thm. A.7, A.14)

Table A.1: Summary of complexity results (new results are in bold).

bounds, we provide as our third contribution a novel polynomial-time reduction from alternating linear bounded automata into the synchronization problems for NFA with partial observability. This is achieved by showing that a combination of the partial observability and nondeterminism can capture alternation. This technique provides matching lower-bounds for all our three synchronization problems on NFA. The lower-bounds for DFA and PFA are derived from the classical problem setting.

In addition, we describe a polynomial-time reduction from a setting with an arbitrary number of observations to an equivalent setting with just two observations, causing only a logarithmic overhead as a factor in the size of the system. Thus all the lower-bound results mentioned earlier remain valid even when restricting the synchronizing strategy synthesis problem to only two observations.

A.1.2 Related Work

The study of synchronizing words initiated by Černý [30] is a variant of our more general strategy synthesis problem where all states return the same observation, and the existence of synchronizing words, short synchronizing words and subset-to-subset synchronizing words have been in different contexts studied up to now; see [117, 122] for recent surveys. The computational complexities of word synchronization problems for DFA, PFA and NFA are summarized in left column of Table A.1. Note that the NLOGSPACE-completeness for the classical synchronization problem on DFA (not explicitly mentioned in the literature) follows directly from the fact that the problem of synchronizing all states is equivalent to checking that any given pair of states can be synchronized [30, 122]. The PSPACE containment of subset-to-subset word synchronization for NFA and PFA follows from [115] by running the algorithm for DFA in an on-the-fly manner, while guessing step-by-step the synchronizing path.

Through the last years there has been an increasing interest in novel settings of the synchronization problem. Volkov et al. [64] study the problem for deterministic automata with positive cost on transitions, and constrain the cost of the synchronizing word. They also study a synchronization game, where the player who wants to synchronize the system proposes only every second character in the synchronizing word. Doyen et al. [51, 50] study the existence of infinite synchronizing words in a probabilistic setting. The theory of synchronizing words have also seen several practical applications, for instance in biocomputing [9], model-based testing [27], and robotics [3].

The notion of homing sequences [99, 67] is related to the study of synchronizing words and to our study of synchronizing strategies. A homing sequence is a sequence of input actions that makes it possible to determine the current state of the system by looking at the outputs from the system. Homing sequences are studied on the model of Mealy machine, essentially a DFA where each transition produces an output from a given finite alphabet (see [117] for a recent survey). Homing sequences have been, among others, studied in an adaptive variant where the next input symbol is determined by the knowledge of the previous outputs. This is related to our synchronizing strategies that depend on the history of observations, however, there are no complexity results for adaptive homing sequence on nondeterministic systems.

Pomeranz and Reddy [108] suggest to combine synchronizing words and adaptive homing sequences. They first apply a homing sequence and then find a word that brings the machine to one particular state. The theory is applied to sequential circuit testing for deterministic systems and their adaptive

synchronization problem can be seen as a subclass of our systems with partial observability (the output actions of a Mealy machine can be encoded as observations).

The idea of gathering the knowledge of possible states where the system can be after a sequence of observable actions, formalized in the notion of knowledge game, is inspired by a similar technique from [40, 28]. Our aggregated knowledge graph technique is related to the super graph construction used in [90]. The complexity of the conditional planning problem from artificial intelligence have also recently been studied under different observability assumptions [114].

Finally, regarding our EXPTIME lower bound, similar complexity results are available for reachability on finite games with partial observability. In [113] the authors study reachability games where both players have only a partial information about the current configuration of the game and show 2-EXPTIME-hardness of deciding whether the first player has a winning strategy. Our synchronization problem for NFA can be also seen as a game, however, here the second player (nondeterminism) has a full information. This variant, called semiperfect-information game, was studied in [41] for a parity objective (including reachability) and the authors show that the problem is both in NP and coNP. Our synchronization problem for NFA is similar to the semiperfect-information game, however, with a very different objective of synchronizing from any given state. This is documented by the fact that the synchronization problem under partial observability for NFA becomes EXPTIME-complete.

A.2 Definitions

We shall now formally rephrase our problem. We define labelled transition systems with partial observability, introduce synchronizing strategies and formulate the three decision problems we are interested in.

Definition A.1. *A* labelled transition system with partial observability *(LTSP) is a quintuple* $T = (S, Act, E, \mathcal{O}, \gamma)$ *where* S *is a set of states,* Act *is an action alphabet,* $E \subseteq S \times Act \times S$ *is the transition relation, written* $s \xrightarrow{a} s'$ *whenever* $(s, a, s') \in E$*,* $\mathcal{O}$ *is a nonempty set of observations, and* $\gamma : S \to \mathcal{O}$ *is a function mapping each state to an observation.*

We shall study the synchronization problems for three natural subclasses of LTSP, namely DFA (deterministic finite automata), PFA (partial finite automata) and NFA (nondeterministic finite automata). An LTSP is called NFA if S, Act and $\mathcal{O}$ are all finite sets. If the transition relation is also

deterministic, i.e. for every $s \in S$ and $a \in Act$ there is at most one $s' \in S$ such that $s \xrightarrow{a} s'$, then we call it PFA. If the transition relation is moreover complete, i.e. for all $s \in S$ and $a \in Act$ there is exactly one $s' \in S$ such that $s \xrightarrow{a} s'$, then we have a DFA. In the rest of the paper we focus on the NFA class and its PFA and DFA subclasses (implicitly assuming partial observability).

For the rest of this section, let $T = (S, Act, E, \mathcal{O}, \gamma)$ be a fixed LTSP. A *path* in T is a finite sequence $\pi = s_1 a_1 s_2 a_2 \dots a_{n-1} s_n$ where $s_i \xrightarrow{a_i} s_{i+1}$ for all i, $1 \leq i < n$. The length of π is the number of transitions, denoted as $|\pi| = n - 1$. The last state s_n in such a path π is referred to as $last(\pi)$. The set of all finite paths in T is denoted by $paths(T)$. The observation sequence of π is the unique sequence of state observations $\gamma(\pi) = \gamma(s_1)\gamma(s_2)\dots\gamma(s_n)$.

A *strategy* on T is a function from finite sequences of observations to next actions to be taken, formally

$$\sigma : \mathcal{O}^+ \to Act \cup \{done\}$$

where $done \notin Act$ is a special symbol signalling that the achieved path is maximal. In the rest of the paper we consider only strategies that are feasible and terminating. A strategy σ is *feasible* if the action proposed by the strategy is executable from the last state of the path; formally we require that for every $\pi = s_1 a_1 s_2 a_2 \dots a_{n-1} s_n \in paths(T)$ that *follows* the strategy, meaning that $\sigma(\gamma(s_1 a_1 s_2 a_2 \dots s_i)) = a_i$ for all i, $1 \leq i < n$, either $\sigma(\gamma(\pi)) = done$ or there is at least one $s' \in S$ such that $last(\pi) \xrightarrow{\sigma(\gamma(\pi))} s'$. A strategy σ is *terminating* if it does not generate any infinite path, in other words there is no infinite sequence $\pi = s_1 a_1 s_2 a_2 \dots$ such that $s_i \xrightarrow{a_i} s_{i+1}$ and $\sigma(\gamma(s_1 a_1 s_2 a_2 \dots s_i)) = a_i$ for all $i \geq 1$.

Given a subset of states $X \subseteq S$ and a feasible, terminating strategy σ, the set of all maximal paths that follow the strategy σ in T and start from some state in X, denoted by $\sigma[X]$, is defined as follows:

$$\begin{aligned}\sigma[X] = \{\pi = s_1 a_1 s_2 a_2 \dots a_{n-1} s_n \in paths(T) \mid s_1 \in X \text{ and } \sigma(\gamma(\pi)) = done \\ \text{and } \sigma(\gamma(s_1 a_1 s_2 a_2 \dots s_i)) = a_i \text{ for all } i,\ 1 \leq i < n \ \} \ .\end{aligned}$$

The set of final states reached when following σ starting from X is defined as $last(\sigma[X]) = \{last(\pi) \mid \pi \in \sigma[X]\}$ and the length of σ from X is defined as $length(\sigma[X]) = \max\{|\pi| \mid \pi \in \sigma[X]\}$. By $length(\sigma)$ we understand $length(\sigma[S])$.

We now define a synchronizing strategy that guarantees to bring the system from any of its states into a single state.

Definition A.2 (Synchronizing strategy)**.** *A strategy σ for an LTSP $T = (S, Act, E, \mathcal{O},$*
$\gamma)$ is synchronizing *if σ is feasible, terminating and $last(\sigma[S])$ is a singleton set.*

Note that synchronizing strategy for NFA means that any execution of the system (for all possible nondeterministic choices) will synchronize into the same singleton set. It is clear that a synchronizing strategy can be arbitrarily long as it relies on the full history of observable actions. We will now show that this is in fact not needed as we can find strategies that do not perform unnecessary steps.

Let $T = (S, Act, E, \mathcal{O}, \gamma)$ be an LTSP and let $\omega \in \mathcal{O}^+$ be a sequence of observations. We define the set of possible states (called belief) where the system can be after observing the sequence ω by

$$belief(\omega) = \{last(\pi) \mid \pi \in paths(T),\ \gamma(\pi) = \omega\}\ .$$

A strategy σ for T is *belief-compatible* if for all $\omega_1, \omega_2 \in \mathcal{O}^+$ with $belief(\omega_1) = belief(\omega_2)$ we have $\sigma(\omega_1) = \sigma(\omega_2)$.

Lemma A.3. *If there is a synchronizing strategy σ for a finite LTSP $T = (S, Act, E,$*
$\mathcal{O}, \gamma)$ then T has also a belief-compatible synchronizing strategy σ' such that $length(\sigma') \leq 2^{|S|}$ and $length(\sigma') \leq length(\sigma)$.

Proof. Let σ be a synchronizing strategy for T. Let $\omega_1, \omega_2 \in \mathcal{O}^+$ be such that $belief(\omega_1) = belief(\omega_2)$. Then the strategy $\sigma_{\omega_1 \to \omega_2}$ defined as

$$\sigma_{\omega_1 \to \omega_2}(\omega) = \begin{cases} \sigma(\omega_2 \omega') & \text{if } \omega = \omega_1 \omega' \\ \sigma(\omega) & \text{otherwise} \end{cases}$$

is also a synchronizing strategy for T; a fact that follows from the definition of $\sigma[X]$ and Definition A.2.

We shall now argue that if there is a synchronizing strategy for T then there is also one of length at most $2^{|S|}$. By contradiction assume that the shortest synchronizing strategy σ for T has length $length(\sigma) > 2^{|S|}$. Among such shortest synchronizing strategies, we pick one that has the smallest number of paths from $paths(T)$ of length $length(\sigma)$. For this strategy σ, there is now a path $\pi = s_1 a_1 s_2 a_2 \dots a_{n-1} s_n \in \sigma(S)$ where $|\pi| = length(\sigma) > 2^{|S|}$ such that $s_1 \in S$, $\sigma(\gamma(\pi)) = done$, and $\sigma(\gamma(s_1 a_1 s_2 a_2 \dots s_i)) = a_i$ for all i, $1 \leq i < n$. Let $\omega = \gamma(\pi)$ be the sequence of observations on this path. As $|\omega| > 2^{|S|}$ and we

have only $2^{|S|}$ possible beliefs, necessarily $\omega = \omega_1\omega_1'$ and $\omega_1 = \omega_2\omega_2'$ such that ω_2' is nonempty and $\mathit{belief}(\omega_1) = \mathit{belief}(\omega_2)$. Then the strategy $\sigma_{\omega_1 \to \omega_2}$ is also a synchronizing strategy for T but has a smaller number of the longest paths (of length $\mathit{length}(\sigma)$) in $\sigma_{\omega_1 \to \omega_2}$ than σ. This contradicts our assumption and we can conclude that the shortest synchronization strategy for T has length at most $2^{|S|}$. Clearly, $\mathit{length}(\sigma_{\omega_1 \to \omega_2}) \leq \mathit{length}(\sigma)$.

Assume now a synchronizing strategy σ such that $\mathit{length}(\sigma) \leq 2^{|S|}$. Then there are only finitely many $\omega_1, \omega_2 \in \mathcal{O}^+$ that can be observed on some path that follows σ such that $\mathit{belief}(\omega_1) = \mathit{belief}(\omega_2)$ and $\sigma(\omega_1) \neq \sigma(\omega_2)$. We can now repeatedly use the strategy substitution $\sigma_{\omega_1 \to \omega_2}$ if $|\omega_2| \leq |\omega_1|$, or $\sigma_{\omega_2 \to \omega_1}$ if $|\omega_1| < |\omega_2|$, in order to construct a belief-compatible strategy for T of length at most $2^{|S|}$ a not longer than $\mathit{length}(\sigma)$. □

We shall now define three versions of the synchronization problem studied in this paper. The first problem simply asks about the existence of a synchronizing strategy.

Problem A.1 (Synchronization)**.** *Given an LTSP T, is there a synchronizing strategy for T?*

The second problem of short-synchronization moreover asks about the existence of a strategy shorter than a given length bound. This can be, for instance, used for finding the shortest synchronizing strategy via the bisection method.

Problem A.2 (Short-Synchronization)**.** *Given an LTSP T and a bound $k \in \mathbb{N}$, is there a synchronizing strategy σ for T such that $\mathit{length}(\sigma) \leq k$?*

Finally, the general subset-to-subset synchronization problem asks to synchronize only a subset of states, reaching not necessarily a single synchronizing state but any state from a given set of final states.

Problem A.3 (Subset-to-Subset Synchronization)**.** *Given an LTSP T and subsets $S_{\mathit{from}}, S_{\mathit{to}} \subseteq S$, is there a feasible and terminating strategy σ for T such that $\mathit{last}(\sigma[S_{\mathit{from}}]) \subseteq S_{\mathit{to}}$?*

If we restrict the set of observations to a singleton set (hence the γ function does not provide any useful information about the current state apart from the length of the sequence), we recover the well-known decision problems studied in the body of literature related to the classical word synchronization (see e.g. [122, 117]). Note that in this classical case the strategy is now simply a fixed finite sequence of actions.

A.3 Complexity Upper-Bounds

In this section we shall introduce the concept of knowledge game and aggregated knowledge graph so that we can conclude with the complexity upper-bounds for the various synchronization problems with partial observability.

A.3.1 Knowledge Game

Let $T = (S, Act, E, \mathcal{O}, \gamma)$ be a fixed LTSP. We define the set of successors from a given state $s \in S$ under the action $a \in Act$ as $succ(s, a) = \{s' \mid s \xrightarrow{a} s'\}$. For $X \subseteq S$ we define

$$succ(X, a) = \begin{cases} \{s' \in succ(s, a) \mid s \in X\} \text{ if } succ(s, a) \neq \emptyset \text{ for all } s \in X \\ \emptyset \qquad\qquad\qquad\qquad \text{otherwise} \end{cases}$$

such that $succ(X, a)$ is nonempty iff every state from X enables the action a. We also define a function $split : 2^S \to 2^{(2^S)}$

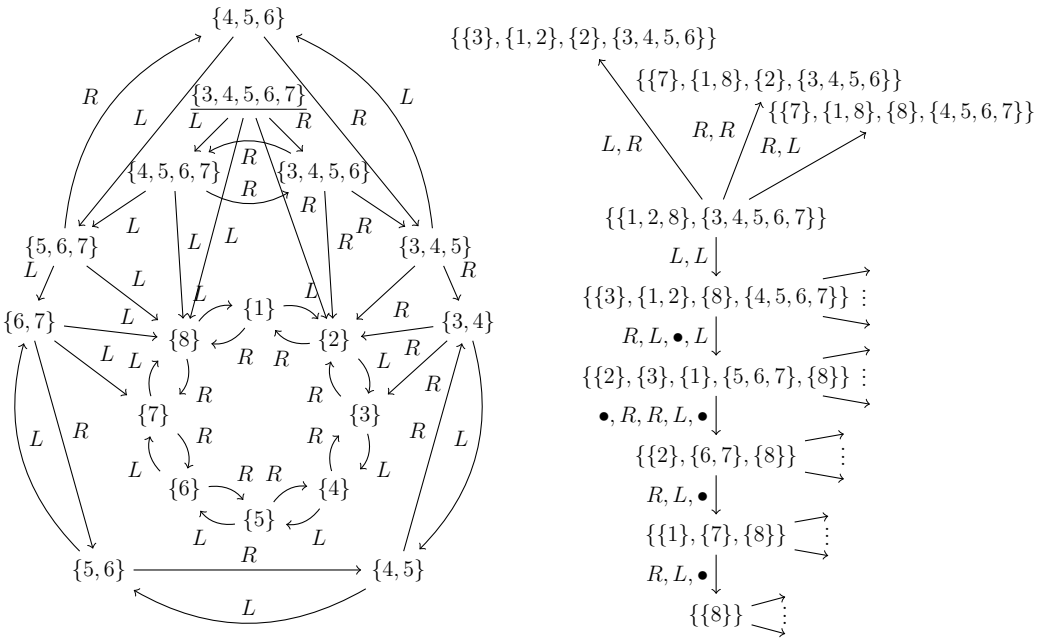
$$split(X) = \{\{s \in X \mid \gamma(s) = o\} \mid o \in \mathcal{O}\} \setminus \emptyset$$

that partitions a given set of states X into the equivalence classes according to the observations that can be made.

We can now define the knowledge game, a two-player game played on a graph where each node represents a belief (a set of states where the players can end up by following a sequence of transitions). Given a current belief, *Player 1* plays by proposing a possible action that all states in the belief can perform. *Player 2* then determines which of the possible next beliefs (partitionings) the play continues from. *Player 1* wins the knowledge game if there is a strategy so that any play from the given initial belief reaches the same singleton belief $\{s\}$. Formally, we define the knowledge game as follows.

Definition A.4. *Given an LTSP $T = (S, Act, E, \mathcal{O}, \gamma)$, the corresponding* knowledge game *is a quadruple $G(T) = (\mathcal{V}, \mathcal{I}, Act, \Rightarrow)$ where*

- $\mathcal{V} = \{V \in 2^S \setminus \emptyset \mid \{V\} = split(V)\}$ *is the set of all unsplittable beliefs,*
- $\mathcal{I} = split(S)$ *is the set of initial beliefs, and*
- $\Rightarrow \subseteq \mathcal{V} \times Act \times \mathcal{V}$ *is the transition relation, written $V_1 \overset{a}{\Rightarrow} V_2$ for $(V_1, a, V_2) \in \Rightarrow$, such that $V_1 \overset{a}{\Rightarrow} V_2$ iff $V_2 \in split(succ(V_1, a))$.*



(a) A knowledge game example rooted at $\{3, 4, 5, 6, 7\}$

(b) A fragment of the aggregated knowledge graph

Figure A.2: Examples of a knowledge game and an aggregated knowledge graph.

Example A.5. *In Figure A.2a we show the knowledge game graph for our running example from Figure A.1a. We only display the part of the graph reachable from the initial belief consisting of states* $\{3, 4, 5, 6, 7\}$ *where the solar panel is inactive. Assume that we want to synchronize from any of these states into the state* 8*. This can be understood as a two-player game where from the current belief* Player 1 *proposes an action and* Player 2 *picks a new belief reachable in one step under the selected action. The question is whether* Player 1 *can guarantee that any play of the game reaches the belief* $\{8\}$*. This is indeed the case and the strategy of* Player 1 *is, for example, to repeatedly propose the action* L *until the belief* $\{8\}$ *is eventually reached.*

We shall now formalize the rules of the knowledge game. A *play* in a knowledge game $G(T) = (\mathcal{V}, \mathcal{I}, Act, \Rightarrow)$ is a sequence of beliefs $\mu = V_1V_2V_3\ldots$ where $V_1 \in \mathcal{I}$ and for all $i \geq 1$ there is $a_i \in Act$ such that $V_i \overset{a_i}{\Rightarrow} V_{i+1}$. The set of all plays in $G(T)$ is denoted $Plays(G(T))$.

A *strategy* (for *Player 1*) is a function $\rho : \mathcal{V} \to Act$. A play $\mu = V_1V_2V_3\ldots$ *follows* the strategy ρ if $V_i \overset{\rho(V_i)}{\Longrightarrow} V_{i+1}$ for all $i \geq 1$. Note that the strategy is memoryless as it depends only on the current belief.

Player 1 wins the game $G(T)$ if there is $s \in S$ and a strategy ρ such that for

every play $\mu = V_1V_2V_3\ldots \in \mathit{Plays}(G(T))$ that follows ρ there exists an $i \geq 1$ such that $V_i = \{s\}$.

The length of a play $\mu = V_1V_2V_3\ldots$ for reaching a singleton belief $\{s\}$ is $\mathit{length}(\mu, s) = i - 1$ where i is the smallest i such that $V_i = \{s\}$. The length of a winning strategy ρ in the game $G(T)$ that reaches the singleton belief $\{s\}$ is

$$\mathit{length}(\rho) = \max_{\mu \in \mathit{Plays}(G(T)),\ \mu \text{ follows } \rho} \mathit{length}(\mu, s)\ .$$

Theorem A.6. *Let $T = (S, Act, E, \mathcal{O}, \gamma)$ and let $G(T) = (\mathcal{V}, \mathcal{I}, Act, \Rightarrow)$ be the corresponding knowledge game where $\mathcal{I} = \mathit{split}(S)$. Then* Player 1 *wins the knowledge game $G(T)$ iff there is a synchronizing strategy for T. Moreover for any winning strategy ρ in the game $G(T)$ there is a synchronizing strategy σ for T such that $\mathit{length}(\rho) = \mathit{length}(\sigma)$, and for any synchronizing strategy σ for T there is a winning strategy ρ in the game $G(T)$ such that $\mathit{length}(\rho) \leq \mathit{length}(\sigma)$.*

Proof. Assume that *Player 1* wins the knowledge game $G(T)$ with the strategy ρ so that all plays reach the belief $\{s\}$. We want to find a synchronizing strategy σ for T. Let the initial observation be $o_1 \in \mathcal{O}$; this gives the initial belief $V_1 = \{t \in S \mid \gamma(t) = o_1\}$. We can now use the winning strategy ρ to determine the first action of our synchronizing strategy $\sigma(o_1) = \rho(V_1)$. By executing the action $\rho(V_1)$, we get the next observation o_2. Now assume that we have a sequence of observations $o_1o_2\ldots o_{i-1}o_i$. We can inductively determine the current belief V_i as

$$V_i = \{t \in \mathit{succ}(V_{i-1}, \rho(V_{i-1})) \mid \gamma(t) = o_i\}$$

for all $i > 1$. This gives us the synchronizing strategy

$$\sigma(o_1o_2\ldots o_{i-1}o_i) = \begin{cases} \mathit{done} & \text{if} \quad V_i = \{s\} \\ \rho(V_i) & \text{otherwise} \end{cases}$$

that guarantees that all plays follow the winning strategy ρ. Hence in any play there exists an $i \geq 1$ such that $V_i = \{s\}$. By this construction it is clear that $\mathit{length}(\rho) = \mathit{length}(\sigma)$.

For the other direction, assume that there is a synchronizing strategy σ for T. Then we know from Lemma A.3 that there exists also a belief-compatible synchronizing strategy σ' where $\mathit{length}(\sigma') \leq \mathit{length}(\sigma)$. We want to find a winning strategy ρ for *Player 1* in $G(T)$. As we know by construction that all states in a belief V have the same observation, we use the notation $\gamma(V) = o$ if $\gamma(t) = o$ for all $t \in V$. Let the initial belief be $V_1 \in \mathcal{I}$. We use the synchronizing

strategy σ' to determine the first action that *Player 1* winning strategy should propose by $\rho(V_1) = \sigma'(\gamma(V_1))$. Now *Player 2* determines the next belief V_2 such that $V_1 \xRightarrow{\sigma'(\gamma(V_1))} V_2$. In general, assume inductively that we reached a belief V_i along the play $\mu = V_1 V_2 \dots V_i$. The winning strategy from V_i is given by

$$\rho(V_i) = \sigma'(\gamma(V_1)\gamma(V_2)\dots\gamma(V_i)) \ .$$

Note that this definition makes sense because σ' is belief-compatible (and hence different plays in the knowledge game that lead to the same belief will propose the same action). From the construction of the strategy and by Lemma A.3 it is also clear that $\mathit{length}(\rho) = \mathit{length}(\sigma') \leq \mathit{length}(\sigma)$. □

We conclude with a theorem proving EXPTIME-containment of the three synchronization problems for NFA (and hence clearly also for PFA and DFA).

Theorem A.7. *The synchronization, short-synchronization and subset-to-subset synchronization problems for NFA are in EXPTIME.*

Proof. We shall first discuss the existence of synchronizing strategy. Let $T = (S, \mathit{Act}, E, \mathcal{O}, \gamma)$ and let $G(T) = (\mathcal{V}, \mathcal{I}, \mathit{Act}, \Rightarrow)$ be the corresponding knowledge game where $\mathcal{I} = \mathit{split}(S)$. By Theorem A.6 there is a synchronizing strategy for T if and only if *Player 1* has a winning strategy in the knowledge game. The game $G(T)$ is of exponential size w.r.t. the system T but we can decide whether *Player 1* has a winning strategy in $G(T)$ in polynomial time (in the size of $G(T)$) by adapting a standard backward reachability algorithm for alternating automata. We mark some singleton belief $\{s\} \in \mathcal{V}$ (we try all possible singleton beliefs if we do not succeed for $\{s\}$). Then we iteratively mark every belief $V \in \mathcal{V}$ such that there is an action $a \in \mathit{Act}$ enabled in V and every V', where $V \overset{a}{\Rightarrow} V'$, is already marked. Once no more beliefs can be marked, we check whether all initial beliefs are marked. If this is the case then there is a winning strategy for *Player 1*. If on the other hand this is not the case for any singleton belief $\{s\}$ then *Player 1* does not have a winning strategy. This gives us an exponential algorithm for the synchronization problem.

Regarding the short-synchronization problem, Theorem A.6 together with Lemma A.3 imply that there is a synchronization strategy in T of length at most k iff there is a winning strategy for *Player 1* in $G(T)$ of length at most k. By modifying the marking algorithm so that it for each marked belief remembers also the length of the shortest path to the singleton belief, we can check whether all initial beliefs are marked such that their length is at most k.

Finally, for the subset-to-subset synchronization problem from the set S_{from} to the set S_{to}, we can modify the game $G(T)$ such that the initial beliefs are

$\mathcal{I} = split(S_{from})$ and a play is winning if it reaches a belief V such that $V \subseteq S_{to}$. The corresponding modification in the marking algorithm is straightforward.

We can so conclude that all three problems are solvable (via the knowledge game graphs) in exponential time w.r.t. to the size of the finite LTSP T. □

A.3.2 Aggregated Knowledge Graph

Knowledge games allowed us to prove EXPTIME upper-bounds for the three synchronization problems on NFA, however, it is in general not possible to guess winning strategies for *Player 1* in polynomial space. Hence we introduce the so-called aggregated knowledge graph where we ask a simple reachability question (one player game). This will provide better complexity upper-bounds for deterministic systems, despite the fact that the aggregated knowledge graph can be exponentially larger than the knowledge game graph.

Definition A.8. *Let $G(T) = (\mathcal{V}, \mathcal{I}, Act, \Rightarrow)$ be a knowledge game. The aggregated knowledge graph is a tuple $AG(G(T)) = (\mathscr{C}, \mathcal{C}_0, \rightrightarrows)$ where*

- *$\mathscr{C} = 2^{\mathcal{V}} \setminus \emptyset$ is the set of configurations (set of aggregated beliefs),*
- *$\mathcal{C}_0 = \mathcal{I}$ is the initial configuration (set of all initial beliefs), and*
- *$\rightrightarrows \subseteq \mathscr{C} \times \mathscr{C}$ is the transition relation such that $\mathcal{C}_1 \rightrightarrows \mathcal{C}_2$, standing for $(\mathcal{C}_1, \mathcal{C}_2) \in \rightrightarrows$, is possible if for every $V \in \mathcal{C}_1$ there is an action $a_V \in Act \cup \{\bullet\}$ such that $V \stackrel{a_V}{\Rightarrow} V'$ for at least one V' (by definition $V \stackrel{\bullet}{\Rightarrow} V$ if and only if $|V| = 1$), ending in $\mathcal{C}_2 = \{V' \mid V \in \mathcal{C}_1 \text{ and } V \stackrel{a_V}{\Rightarrow} V'\}$.*

Example A.9. *Figure A.2b shows a fragment of the aggregated knowledge graph for our running example from Figure A.1a. The initial configuration is the aggregation of the initial beliefs and each transition is labelled with a sequence of actions for each belief in the aggregated configuration. The suggested path shows how to synchronize into the state 8. Note that the action $\bullet$, allowed only on singleton beliefs, stands for the situation where the belief is not participating in the given step.*

Theorem A.10. *Let $G(T) = (\mathcal{V}, \mathcal{I}, Act, \Rightarrow)$ be a knowledge game and let $AG(G(T)) = (\mathscr{C}, \mathcal{C}_0, \rightrightarrows)$ be the corresponding aggregated knowledge graph. Then $\mathcal{C}_0 \rightrightarrows^* \{\{s\}\}$ for some state s if and only if* Player 1 *wins the knowledge game $G(T)$. Moreover, for any winning strategy ρ in $G(T)$ that reaches the singleton belief $\{s\}$ we have $\mathcal{C}_0 \rightrightarrows^{length(\rho)} \{\{s\}\}$, and whenever $\mathcal{C}_0 \rightrightarrows^n \{\{s\}\}$ then there is a winning strategy ρ in $G(T)$ such that $length(\rho) \leq n$.*

Proof. Assume that for some s the path $\eta = \mathcal{C}_0 \rightrightarrows^n \{\{s\}\}$ exists in the aggregated knowledge graph $AG(G(T))$. Then there is also a path $\eta' = \mathcal{C}_0 \rightrightarrows^{n'} \{\{s\}\}$ such that for each belief we always propose the same action, no matter what configuration it is in. This follows by the argument that we can use the same strategy for each belief that appears in the aggregated knowledge graph by following the one that makes the path shortest (reaches the belief $\{s\}$ faster). Clearly $|\eta'| \leq |\eta|$. Now we construct a winning strategy ρ for *Player 1* in the knowledge game $G(T)$ such that $\rho(V) = a_V$ where $V \in C \in \eta'$ and a_V is the proposed action for the belief V in the configuration C. This is a winning strategy for *Player 1* as it reaches the belief $\{s\}$. By construction of the strategy it is also clear that $\mathit{length}(\rho) \leq n$.

Now in the other direction, assume that *Player 1* wins the knowledge game $G(T)$ with the strategy ρ by bringing all plays to the belief $\{s\}$ where $s \in S$. We want to show that there is a path $\mathcal{C}_0 \rightrightarrows^* \{\{s\}\}$ in aggregated knowledge graph $AG(G(T))$. We start with the initial configuration $\mathcal{C}_0 = \{V_0 \mid V_0 \in \mathcal{I}\}$. Assume that we already have a prefix of the path up to the configuration $\mathcal{C}$. The next configuration is then

$$\mathcal{C}' = \{V' \mid V \in \mathcal{C} \text{ and } V \overset{a_V}{\Longrightarrow} V' \text{ where } a_V = \rho(V) \\ \text{if } V \neq \{s\} \text{ and } a_V = \bullet \text{ otherwise } \} \ .$$

It is clear that this will bring us to the configuration $\{\{s\}\}$ as all beliefs now follow the winning strategy ρ. By counting the number of steps in the path, we get that $\mathcal{C}_0 \rightrightarrows^{\mathit{length}(\rho)} \{\{s\}\}$. □

The aggregated knowledge graph can in general be exponentially larger than its corresponding knowledge game as the nodes are now subsets of beliefs (that are subsets of states). Nevertheless, we can observe that for DFA and PFA, the size of configurations in $AG(G(T))$ cannot grow.

Lemma A.11. *Let T be an LTSP generated by DFA or PFA. Let $AG(G(T)) = (\mathscr{C}, \mathcal{C}_0, \rightrightarrows)$ be the corresponding aggregated knowledge graph. Whenever $C \rightrightarrows C'$ then $\sum_{V \in C} |V| \geq \sum_{V' \in C'} |V'|$.*

Proof. Let $V \in C$ and let $X = \{V' \mid V \overset{a_V}{\Longrightarrow} V'\}$. It is now enough to argue that $|V| \geq \sum_{V' \in X} |V'|$. Recall that $V' \in X$ iff $V' \in \mathit{split}(\mathit{succ}(V, a_V))$. Clearly $|\mathit{succ}(V, a_V)| \leq |V|$ for DFA and PFA. The rest follows from the fact that the function *split* only partitions $\mathit{succ}(V, a_V)$ and hence does not increase its size. □

Theorem A.12. *The synchronization, short-synchronization and subset-to-subset synchronization problems for DFA and PFA are decidable in* PSPACE.

Proof. By Theorem A.10 and Theorem A.6 we get that we can reach the configuration $\{\{s\}\}$ for some $s \in S$ in the aggregated graph $AG(G(T))$ if and only if there is a synchronizing strategy for the given LTSP T. From Lemma A.11 we know that for DFA and PFA the size of each aggregated configuration reachable during any computation is bounded by the size of the set S and therefore can be stored in polynomial space. As PSPACE is closed under nondeterminism, the path to the configuration $\{\{s\}\}$ for some $s \in S$ can be guessed, resulting in a polynomial-space algorithm for the synchronizing problem. Theorem A.10 also implies that the length of the shortest synchronizing strategy in T is the same as the length of the shortest path to the configuration $\{\{s\}\}$ for some s, giving us that the short-synchronization problems for DFA and PFA are also in PSPACE. Regarding the subset-to-subset synchronization problem from the set S_{from} to the set S_{to}, we can in a straightforward manner modify the aggregated knowledge graph so that the initial configuration is produced by splitting S_{from} according to the observations and we end in any configuration consisting solely of beliefs V that satisfy $V \subseteq S_{to}$ (while allowing the action $\bullet$ from any such belief to itself). □

Finally, for the synchronization and short-synchronization problems on DFA, we can derive even better complexity upper-bounds by using the aggregated knowledge graph.

Theorem A.13. *The synchronization problem on DFA is in* NLOGSPACE *and the short-synchronization problem on DFA is in* NP.

Proof. Using the same arguments as in Theorem A.12, we can derive that the question of synchronizing two given states s_1 and s_2 can be done in nondeterministic logarithmic space using the aggregated knowledge graph (by Lemma A.11 we know that on any path in the aggregated graph we need to remember at most two states). Observe now that we can synchronize all pairs of states independently if and only if all states can be synchronized at once (a straightforward generalization of the result from the classical setting without partial observability [30, 122]).

For the containment of the short-synchronization problem in NP, we first notice that if n states can be synchronized then the shortest synchronizing strategy has length at most $(n-1)n^2$. This follows from the fact that each pair-wise synchronization takes at most n^2 steps (in the aggregated knowledge graph with configurations containing just two states we get a loop if the length of the path exceeds n^2). Clearly synchronizing one state takes zero steps. Now by induction, assume that synchronizing the first i states into a state t requires at most $(i-1)n^2$. Let s be $(i+1)$'th state that we should also synchronize

with. From the initial state s, we follow the strategy for synchronizing the first i states, arriving into a state s'. Now we extend the strategy so that we pairwise synchronize t and s', taking at most n^2 steps as argued above. Hence to synchronize $i+1$ states we need at most $(i-1)n^2 + n^2 = in^2$ steps, providing us with the conclusion that synchronizing n states requires a strategy of length at most $(n-1)n^2$. Such a strategy can be guessed and verified in nondeterministic polynomial time. □

A.4 Complexity Lower-Bounds

We shall now describe a technique that will allow us to argue about EXPTIME-hardness of the synchronization problems for NFA.

First, we introduce the acceptance problem of alternating linear bounded automata that is known to be EXPTIME-complete (see e.g. [118]). For technical convenience, we present a less standard variant of the problem where only internal control states can read/write on the tape while the existential and universal states that enable branching do not modify or even know the tape content. The presented variant has the same expressive power as the standard model.

An alternating linear bounded automaton (ALBA) is a tuple $\mathcal{M} = (Q_i, Q_\forall, Q_\exists, Q, \Sigma,$
$q_0, q_{acc}, \vdash, \dashv, \delta_1, \delta_2)$ where

- Q_i is a set of internal control states,
- $Q_\forall$ is a set of universal control states,
- $Q_\exists$ is a set of existential control states,
- $Q = Q_i \uplus Q_\exists \uplus Q_\forall$ is the set of all control states,
- $\Sigma = \{a, b\}$ is the input alphabet,
- $q_0 \in Q$ is the initial state,
- $q_{acc} \in Q$ is the accepting state,
- $\delta_1 : Q_i \times (\Sigma \cup \{\vdash, \dashv\}) \to Q \times (\Sigma \cup \{\vdash, \dashv\}) \times \{L, R\}$ is a transition relation for internal states such that
  - $\delta_1(q, \vdash) = (q', \vdash, R)$,
  - $\delta_1(q, \dashv) = (q', \dashv, L)$,

  - $\delta_1(q, x) = (q', x', D)$,

  where $q \in Q_i$, $q' \in Q$, $x, x' \notin \{\vdash, \dashv\}$ and $D \in \{L, R\}$,

- $\delta_2 : (Q_\exists \cup Q_\forall) \to Q \times Q$ is a transition relation for existential and universal states.

For each $q \in Q_\forall \cup Q_\exists$ the transition relation $\delta_2(q)$ returns a pair of two elements (q_1, q_2), referred to as the *first* and *second* successor, respectively.

A *configuration* of an ALBA $\mathcal{M}$ is the current state, the position of the head on the tape and the tape content (starting and ending with the end-markers). We denote configurations as $w_1 q w_2$ where $q \in Q$ is the current state, and $w_1 = \vdash w_1'$ and $w_2 = w_2' \dashv$ where $w_1', w_2' \in \Sigma^*$ represent the tape content such that the head points to the first letter of w_2. The *initial configuration* for the input word w is $c_0 = \vdash q_0 w \dashv$. Depending on the control state a configuration is called *internal* if $q \in Q_i$, *existential* if $q \in Q_\exists$, *universal* if $q \in Q_\forall$ and *accepting* if $q = q_{acc}$.

A *step of computation* is a relation $\to$ between configurations defined as follows (where $x, x', y \in \Sigma \cup \{\vdash, \dashv\}$, $w_1, w_2 \in (\Sigma \cup \{\vdash, \dashv\})^*$ such that all configurations start with $\vdash$ and end with $\dashv$):

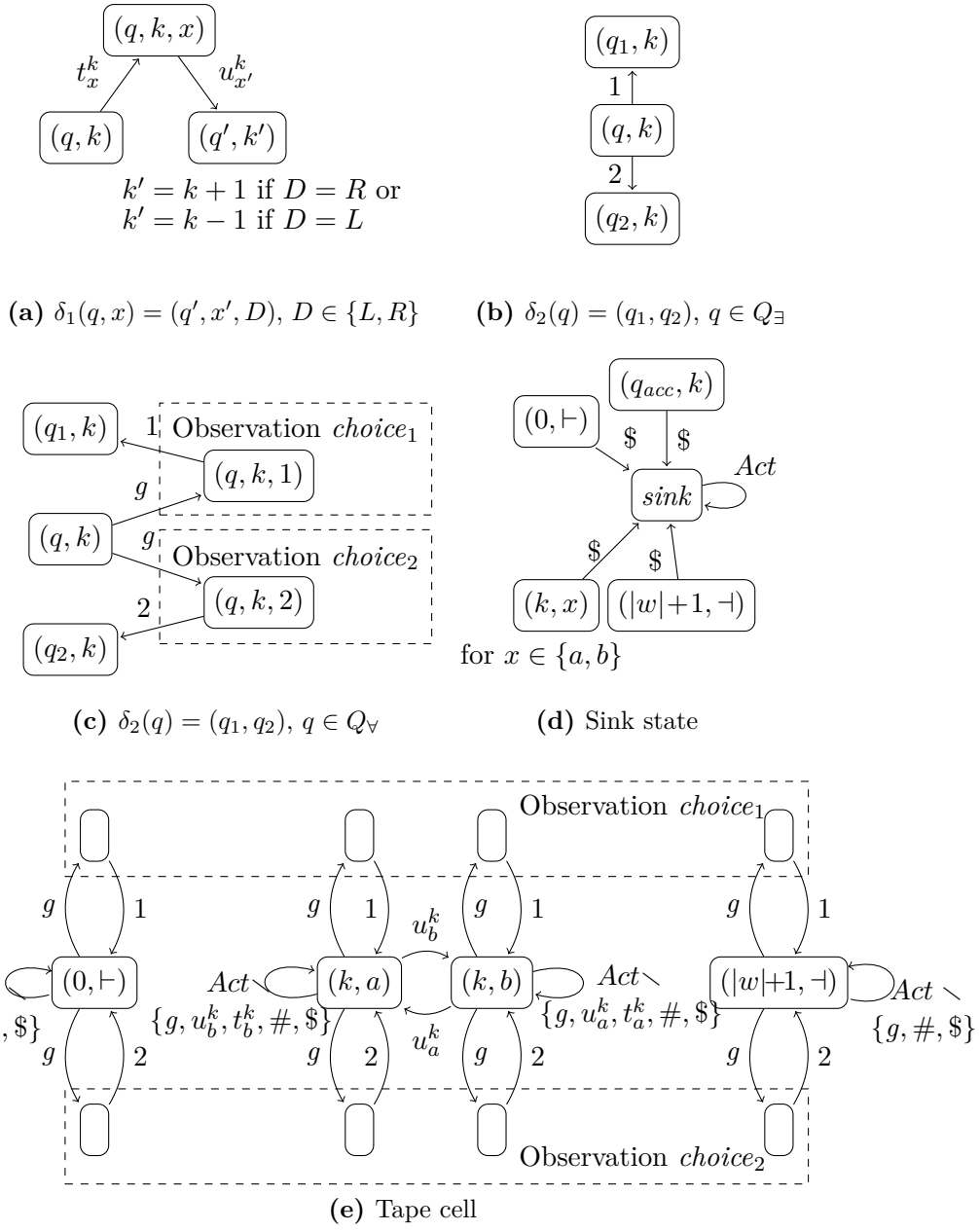
- $w_1 q x w_2 \ \to \ w_1 x' q' w_2$ whenever $q \in Q_i$ and $\delta_1(q, x) = (q', x', R)$,
- $w_1 y q x w_2 \ \to \ w_1 q' y x' w_2$ whenever $q \in Q_i$ and $\delta_1(q, x) = (q', x', L)$, and
- $w_1 q w_2 \to w_1 q' w_2$ whenever $q \in Q_\exists \cup Q_\forall$, $\delta_2(q, x) = (q_1, q_2)$ and $q' = q_1$ or $q' = q_2$.

A *computation tree* for $\mathcal{M}$ on an input $w \in \Sigma^*$ is a possibly infinite configuration-labelled tree rooted with the initial configuration $c_0 = \vdash q_0 w \dashv$ such that every node labelled with a configuration c satisfies:

- if c is accepting then it is a leaf,
- if c is internal or existential then it has one child c' such that $c \to c'$, and
- if c is universal then it has two children labelled with the first and the second successor of c.

An ALBA $\mathcal{M}$ *accepts* a string $w \in \{a, b\}^*$ iff there is a finite computation tree for $\mathcal{M}$ on w such that all leaves are accepting configurations. It is well known that the problem whether an ALBA accepts a string w is EXPTIME-complete.

Theorem A.14. *The subset-to-subset synchronization problem is EXPTIME-hard for NFA.*



(a) $\delta_1(q,x) = (q',x',D)$, $D \in \{L,R\}$ **(b)** $\delta_2(q) = (q_1,q_2)$, $q \in Q_\exists$

(c) $\delta_2(q) = (q_1,q_2)$, $q \in Q_\forall$ **(d)** Sink state

(e) Tape cell

Figure A.3: Encoding of the δ-function (k ranges over all its possible values).

Proof. We shall now provide a polynomial-time reduction from the acceptance problem for ALBA to synchronization problems on NFA with partial observability. Assume a given ALBA $\mathcal{M} = (Q_i, Q_\forall, Q_\exists, Q, \Sigma, q_0, q_{acc}, \vdash, \dashv, \delta_1, \delta_2)$ and a string w. We construct a finite LTSP (NFA) $T = (S, Act, E, \mathcal{O}, \gamma)$ where

- $S = \{sink\} \cup \{(q,k) \mid q \in Q,\ 0 \leq k \leq |w|+1\} \cup$
  $\{(q,k,a),(q,k,b) \mid q \in Q_i,\ 1 \leq k \leq |w|\} \cup$
  $\{(q,0,\vdash),(q,|w|+1,\dashv)\} \cup$
  $\{(q,k,1),(q,k,2) \mid q \in Q_\forall,\ 0 \leq k \leq |w|+1\} \cup$
  $\{(k,x),(k,x,1),(k,x,2) \mid 1 \leq k \leq |w|,\ x \in \{a,b\}\} \cup$
  $\{(0,\vdash),(0,\vdash,1),(0,\vdash,2)\} \cup$
  $\{(|w|+1,\dashv),(|w|+1,\dashv,1),(|w|+1,\dashv,2)\}$,
- $Act = \{t_x^k, u_x^k \mid 1 \leq k \leq |w|,\ x \in \{a,b\}\} \cup$
  $\{t_\vdash^0, u_\vdash^0, t_\dashv^{|w|+1}, u_\dashv^{|w|+1}\} \cup \{g, 1, 2, \$, \#\}$,
- $\mathcal{O} = \{default, choice_1, choice_2\}$, and
- the transition relation E together with γ is given in Figure A.3. Note that all states that are not marked with the observations $choice_1$ or $choice_2$, are assigned by default the observation *default*.

The construction above provides a reduction to the subset-to-subset synchronization problem where we assume that we synchronize from the initial states

- $(q_0, 1)$ and
- (k, x_k) for all k, $0 \leq k \leq |w|+1$, such that the input word w is of the form $x_1 x_2 \ldots x_n$ and $x_0 = \vdash$ and $x_{n+1} = \dashv$

into the set $\{sink\}$. During the simulation of the given ALBA, we shall preserve the invariant that there is exactly one active state of the form (q,k) representing that we are at the control state q and the head is at position k. Also for every tape cell at position k, we remember the current symbol stored in each cell by being either in the state (k,a) or (k,b) (with the exception of the end-markers that can store only one symbol).

Consider now that the active control state is (q,k). There are four cases according to whether q is internal, existential, universal or accepting control state.

- Let $q \in Q_i$. The corresponding transitions are depicted in Figure A.3a. The state (q,k) can perform the test action t_a^k or t_b^k depending on whether the k'th tape cell (Figure A.3e) is in the state (k,a) or (k,b), respectively. Choosing a wrong test action means that the tape cell cannot perform the chosen action, implying that this may not be a synchronizing strategy.

After the appropriate test action was chosen, we have to necessarily perform the update action $u^k_{x'}$ bringing us to the state (q', k') where the head is moved accordingly and this action has to synchronize with the k'th tape cell so that the new tape symbol x' is updated accordingly. Notice that any other tape cell, save the one on k'th position, simply mimics these two actions via self-loops in their current states.

- Let $q \in Q_\exists$. The corresponding transitions are depicted in Figure A.3b. Here we can freely choose the successor control state by pick the action 1 or 2 according to the first and second successor given by the δ_2 function. Clearly, all tape cells will mimic the chosen action using a self-loop.
- Let $q \in Q_\forall$. The corresponding transitions are depicted in Figure A.3c. After performing the guessing action g, the system nondeterministically decides whether to enter $(q, k, 1)$ or $(q, k, 2)$ and we have to investigate the possible continuation from both situations. However, as they are in different observation classes, we can split our strategy and design different continuations for these two possibilities. The main point is now about the tape cells in Figure A.3e. They have also a nondeterministic choice about going to state with observation $choice_1$ or $choice_2$ but they do not have to follow the control state choice. However, if they do not follow it, we can observe such a behavior and design an alternative strategy for the tape cell, continuing the simulation like if the opposite choice was taken in the control states.
- Let $q = q_{acc}$. Then the transitions in Figure A.3d apply and we can move using the action \$ into a global state called *sink* that is the only state that allows to synchronize both control states and tape cells. Clearly, any tape cell is able to perform \$ and enter the synchronizing state *sink* at any time, but only the accepting control state is able to enter the sink state.

This completes the proof and we have show that the constructed subset-to-subset synchronization problem has a synchronizing strategy if and only if the given ALBA accepts the input string, giving us the following hardness result. □

Theorem A.15. *The synchronization and short-synchronization problems are*
EXPTIME-hard for NFA.

Proof. In order to prove EXPTIME-hardness for the existence of synchronization strategy from any given initial state, we need to introduce additional transitions

together with a new state *init* as depicted in Figure A.4. These transitions add a new action # in such a way that any synchronizing strategy has to start by performing the action #. If any other action should be chosen instead of # then it is impossible to synchronize the state *init*. It is now clear that performing this initialization brings the system to the set of initial states in the subset-to-subset problem discussed above, deriving the following theorem. Note that for the short-synchronization case, we use Lemma A.3 giving us an exponential upper-bound on the length of the shortest synchronizing strategy. □

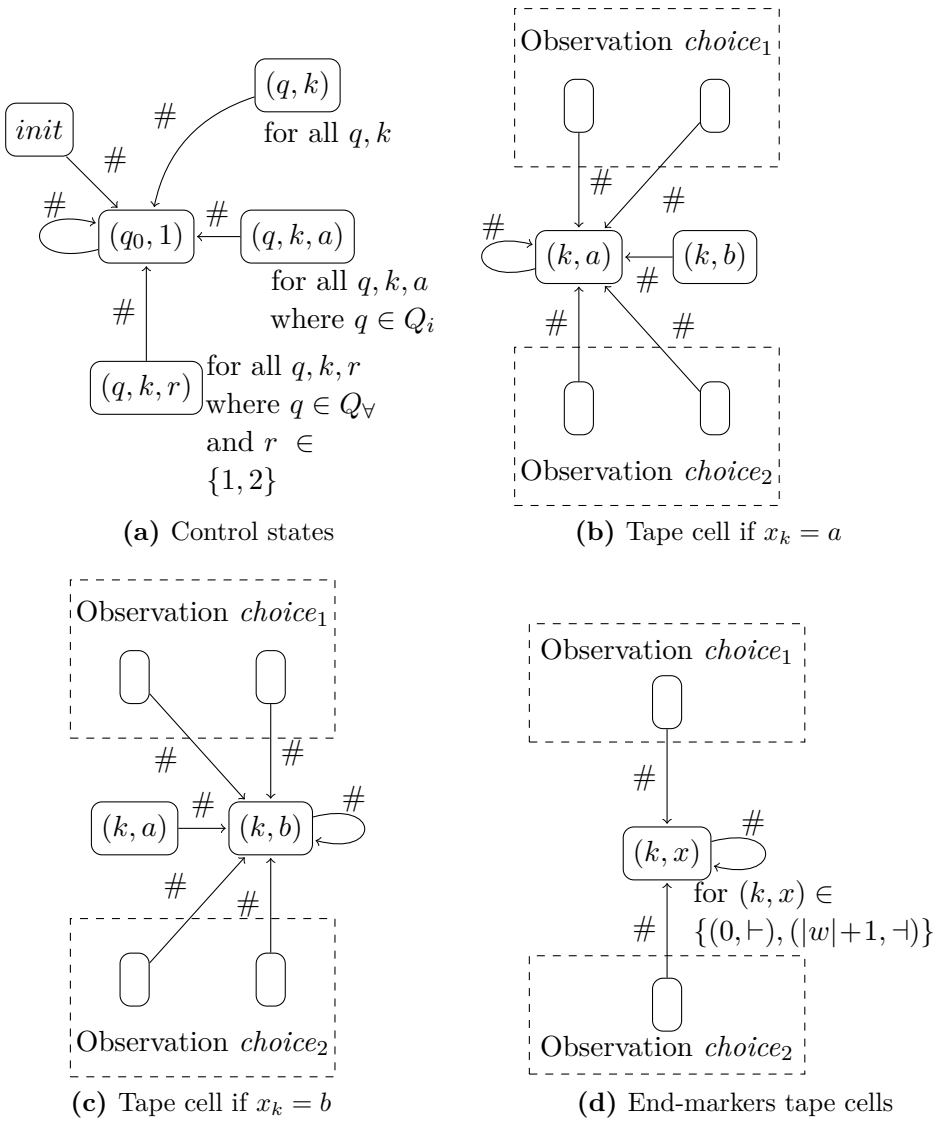
The reader may wonder whether three different observations are necessary for proving EXPTIME-hardness of the synchronizing problems or whether one can show the hardness only with two. By analysis of the construction, we can observe that two observations are in fact sufficient. Moreover, there is a general polynomial-time reduction from a given synchronization problem with an arbitrary number of observations to just two observations, while increasing the size of the system by only a logarithmic factor.

Theorem A.16. *The synchronization, short-synchronization and subset-to-subset synchronization problems on DFA, PFA and NFA are polynomial-time reducible to the equivalent problems with only two observations.*

Proof. Let $T = (S, Act, E, \mathcal{O}, \gamma)$ be a given finite LTSP and let $\ell = \lceil \log |\mathcal{O}| \rceil$. We can assume that all observations from $\mathcal{O}$ are written in binary and contain exactly ℓ bits (including the leading zeros). Then the i'th bit of an observation $o \in \mathcal{O}$ is denoted as o^i. We construct an LTSP $T' = (S', Act', E', \mathcal{O}', \gamma')$ such that

- $S' = S \cup \{s_0, s_1, s_2, \ldots, s_{\ell-1} \mid s \in S\}$,
- $Act' = Act \cup \{\bullet, \#\}$,
- $E' = \{(s, a, s_1) \mid s' \xrightarrow{a} s\} \cup \{(s_1, \bullet, s_2), (s_2, \bullet, s_3), \ldots, (s_{\ell-2}, \bullet, s_{\ell-1}), (s_{\ell-1}, \bullet, s) \mid s \in S\} \cup \{(s_1, \#, s_1), (s_2, \#, s_1), (s_3, \#, s_1), \ldots, (s_{\ell-1}, \#, s_1), (s, \#, s_1) \mid s \in S\} \cup \{(s_i, a, s_i) \mid s \in S,\ 1 \leq i < \ell,\ a \in Act \setminus \{\bullet, \#\}\} \cup \{(s, \bullet, s) \mid s \in S\} \cup \{(s_0, a, s_1) \mid s \in S,\ a \in Act\}$,
- $\mathcal{O}' = \{0, 1\}$, and
- $\gamma'(s) = \gamma(s)^\ell$ and $\gamma'(s_i) = \gamma(s)^i$ for all $s \in S$ and all i, $1 \leq i < \ell$, and $\gamma'(s_0) = 0$ for all $s \in S$.

The construction is depicted in Figure A.5. The main idea is now that instead of entering the state s in the original system, we enter the newly added state



(a) Control states **(b)** Tape cell if $x_k = a$

(c) Tape cell if $x_k = b$ **(d)** End-markers tape cells

Figure A.4: Initialization for the input $w = x_1 x_2 \dots x_n$ by adding a new state *init*.

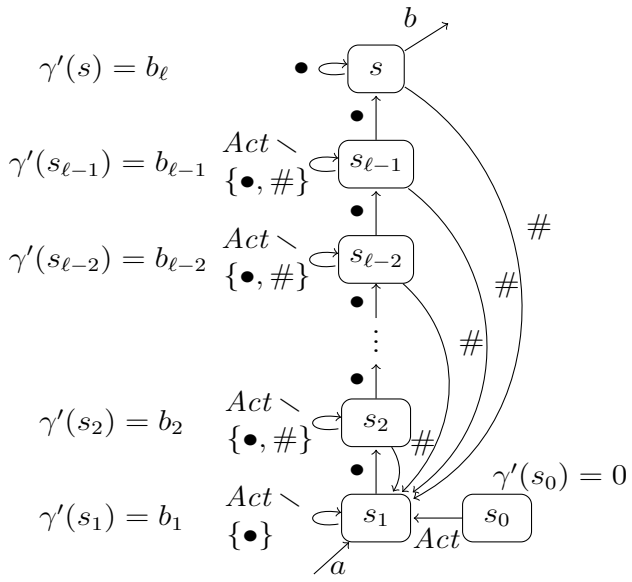


Figure A.5: New states for every $s \in S$ where $\gamma(s) = b_1 b_2 \dots b_\ell$; the arrow labelled with a represents incoming transitions to s and the one labelled with b outgoing transitions from s.

s_1 associated with s. Now by performing the sequence of actions $\bullet$, we obtain step-by-step the knowledge about the observation in the state s. The added self-loops are necessary only for the case of DFA in order to have a complete transition relation and the states s_0 (not reachable from any other states) are important only for arguing about the length of the synchronizing strategy. Now we can show that the synchronization, short-synchronization and subset-to-subset synchronization problems have a solution in T iff they have a solution in T'.

Assume a synchronizing strategy in the original LTSP T. The equivalent strategy in T' will simply initially perform the action # so that is gets the possibility to read the whole information about the observation in the given state. After this the actions in the strategy T are separated by ℓ-1 actions $\bullet$ in order to obtain a strategy for T'. Such a sequence provides a full information about the original observation in the state s, allowing us to follow faithfully the given synchronizing strategy from T also in T'.

On the other hand, the system T' does not provide any additional information by performing # compared with T; note that after # the system T' must perform the actions $\bullet$ until reaching the original state s as it is impossible to synchronize the system in the newly added states $s_0, \dots, s_{\ell-1}$ and exercising the self-loops in the newly added states does not help either. Hence any synchronizing strategy in T' can be transformed into a synchronizing strategy

in T by leaving out the intermediate actions $\bullet$ and $\#$.

For the short-synchronization problem for T, checking if the length of the strategy is at most k, we will instead ask in T' whether there is a strategy of length at most $k \cdot (\ell - 1) + 1$. □

A.5 Conclusion

A summary of results is provided in Table A.1, referring explicitly to theorems with the corresponding claims. It is clear that all complexity lower-bounds for the classical word synchronization transfer to the more general setting with partial observability. We were able to match those lower-bounds with corresponding upper-bounds for the cases of DFA and PFA where the transition relation is deterministic. In case of nondeterministic systems, the three synchronization problems moved to a higher complexity class (even for systems with just two observations) as the combination of nondeterminism with partial observability allows us to encode alternation.

In the future work, we plan to study more general synchronization problems where the sequences of synchronizing actions can be restricted by a logical formula (requiring for example a certain ordering of actions), or we may want to optimize synchronizing strategies from a quantitative point of view (find the cheapest synchronizing strategy).

Paper B

Polynomial Time Decidability of Weighted Synchronization under Partial Observability

Simon Laursen Kim G. Larsen Jiří Srba
Aalborg University, Department of Computer Science, Denmark

Jan Křetínský
ITS Austria

Abstract We consider weighted automata with both positive and negative integer weights on edges and study the problem of synchronization using adaptive strategies that may only observe whether the current weight-level is negative or nonnegative. We show that the synchronization problem is decidable in polynomial time for deterministic weighted automata.

Publication History This paper [89] has been published in the Proceedings of the 26th International Conference on Concurrency Theory, (CONCUR 2015), in LIPIcs volume 42, page 142–154, by Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015. This thises includes a full version of the paper with all proves and a modified layout.

The layout has been revised.

B.1 Introduction

The problem of synchronizing automata [30, 117, 122] studies the following natural question: *"how can we gain control over a device when its current state is unknown?"* Synchronizing automata have classically been studied in the setting of deterministic finite automata (DFA), aiming at finding short synchronizing words, i.e. finite sequences of input symbols that will bring the automaton from any (unknown) state into a unique state. Here the existence of a synchronizing word is NLOGSPACE-complete [30, 122], and polynomial bounds were given on the length of the shortest synchronizing word. Yet, establishing a tight bound on the length of the shortest synchronizing word has been an open problem for the last 50 years, with Černý [30] conjecturing that words of length at most $(n-1)^2$, where n is the number of states in the DFA, are sufficient.

We consider synchronization of deterministic weighted automata (WA), where their states are composed of locations and integer weights, and where transitions have their associated weights from $\mathbb{Z}$. In this setting, weights are simply accumulated during the run of the system, and thus it is impossible to find a word that will ensure synchronization to a single state: for any two states with identical locations but different weights, e.g. (ℓ, z) and $(\ell, z+1)$, any word will—by the assumption of determinism—maintain the relative difference in their weights. We therefore assume that during the synchronization, the controller has some (minimal) information available concerning the current weight of the system; in particular, we assume that the controller is able to observe whether the current weight is negative or nonnegative. Under this assumption, a solution to the synchronization problem becomes an *adaptive* strategy, in the sense that the next input to be selected may be based on the previous weight-observations made by the controller.

Our main result is that the existence of a synchronizing strategy, using only observations of the sign of the current weight-level, is decidable in polynomial time for deterministic WA. This result relies on a polynomial time algorithm for detecting cycles of weight $+1$ and -1 in a given weighted graph.

Fig. B.1 illustrates BP (Blind Packman), a WA with 6 locations and 4 actions. We have to find a strategy that will (under partial observability of weights) synchronize infinitely many states of the form $(\ell_{i,j}, z)$ where $\ell_{i,j}$ is one of the 6 locations and $z \in \mathbb{Z}$ is the starting weight. First, we note that after an n-input, BP will be in one of the 3 (top-row) locations $\ell_{i,2}$ for $i = 1, 2, 3$. Given the cyclic, horizontal structure, it is also clear that no further sequence of inputs from the set $\{\mathsf{n}, \mathsf{e}, \mathsf{s}, \mathsf{w}\}$ can provide any additional information in which of the

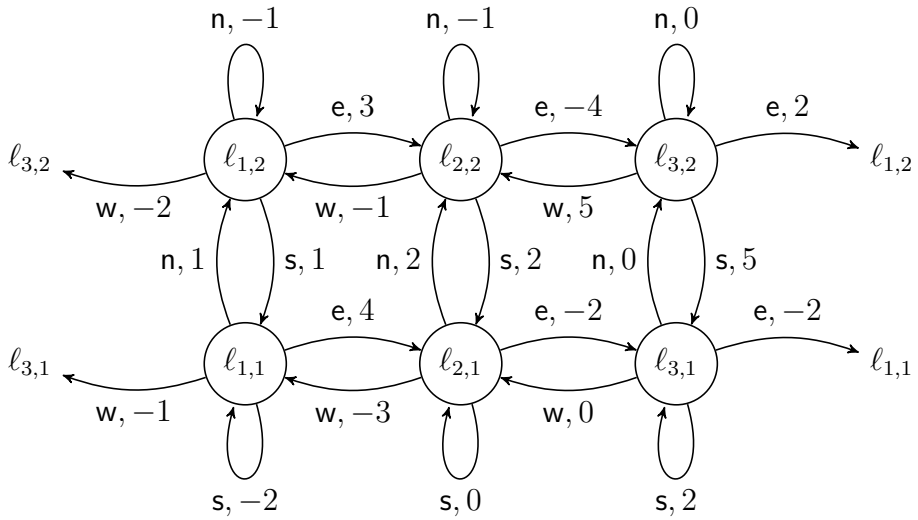


Figure B.1: **B**lind **P**ackman with moves north (n), south (s), east (e) and west (w)

three locations we are located. However, assuming the weight-level is observed to be nonnegative (a similar case applies if the weight-level is observed to be negative), we may infer that BP is in a state of the form $(\ell_{i,2}, z)$ with $z \geq 0$. Noting the -1-loops from $\ell_{1,2}$ and $\ell_{2,2}$, it is tempting to repeatedly offer n as input until the weight-level becomes negative. However, the presence of the 0-loop at $\ell_{3,2}$ makes it possible that such a strategy will not terminate. Instead, we observe that the input-word $(\mathsf{n} \cdot \mathsf{e})^3$ will constitute a (composite) cycle in BP that makes the weight-level drop exactly by -1, regardless as to from which of the three top-locations the word was executed. Thus, by repeating this input-word while constantly observing the sign of the weight-level and terminating as soon as the weight-level becomes negative, we are able to infer that the BP automaton is either in the location $\ell_{3,2}$ in case the observation changed after the input e, or in one of the states $(\ell_{1,2}, -1)$ or $(\ell_{2,2}, -1)$ if the change happened after the input n. In the former case, we exercise the cycle e^3 until a change in observation brings us to $(\ell_{3,2}, 0)$. In the latter case, an e-input followed by a test of the weight-level will reveal the true identity of the state; using ± 1 cycles, it is now easy to reach $(\ell_{3,2}, 0)$, thus completing the synchronization.

As illustrated by the above example, the presence of cycles with weights $+1$ and -1 is essential for the synchronization under partial observability. As we shall demonstrate, the existence of such cycles is decidable in polynomial time, and constitutes, with a few other polynomial time checks, a necessary and sufficient condition for the synchronization of a WA.

Related Work

Survey of results and applications for classical synchronizing words may be found in [117, 122]. Recently, there has been an increasing interest in novel extensions of the synchronization problem. Volkov [64] studied synchronization games and priced synchronization on weighted automata with positive weights and finds a synchronizing word where the worst accumulated cost is below a given bound. In [50, 51, 53], (infinite) synchronizing words were studied in the probabilistic settings. Synchronization of weighted timed automata was studied in [49], where location synchronization with safety conditions on the weight was considered, though without the requirement on the weight synchronization. Finally, synchronization under partial observability was recently studied in [95] but only in the context of finite automata without weights.

B.2 Definitions

We shall now formally define the synchronization problem on deterministic and complete weighted automata.

Definition B.1 (Weighted Automaton)**.** *A (deterministic)* weighted automaton (WA) *is a tuple* $\mathcal{A} = (L, Act, E, W)$ *where*

- L *is a finite set of locations,*
- Act *is a finite set of actions,*
- $E : L \times Act \to L$ *is a transition function, and*
- $W : L \times Act \to \mathbb{Z}$ *is a weight function.*

A *state* of $\mathcal{A}$ is a pair $(\ell, z) \in L \times \mathbb{Z}$ where ℓ is the current location and z the current weight. Let $S(\mathcal{A})$ be the set of all states of $\mathcal{A}$. We write $(\ell, z) \xrightarrow{a,w} (\ell', z')$ if $E(\ell, a) = \ell'$, $W(\ell, a) = w$ and $z' = z + w$.

A *path* in $\mathcal{A}$ is a finite sequence of states $\pi = s_0 s_1 \dots s_n$ such that for all i, $0 \leq i < n$, we have $s_i \xrightarrow{a_i,w_i} s_{i+1}$ for some $a_i \in Act$ and $w_i \in \mathbb{Z}$. The last state s_n in the path π is referred to as $last(\pi)$. The set of all paths is denoted by $paths(\mathcal{A})$. For the complexity analysis in the rest of this paper, we assume a binary encoding of integers in $\mathcal{A}$.

Definition B.2 (Observation Function)**.** *An* observation function $\gamma : S(\mathcal{A}) \to \mathcal{O}$ *maps each state of* $\mathcal{A}$ *to an observation from an observations set* $\mathcal{O}$*.*

Assume now a given observation function γ to the set of observations $\mathcal{O}$. Let $\pi = s_0 s_1 \dots s_n$ be a path in $\mathcal{A}$. The observation function γ is naturally extended to an *observation sequence* for π by

$$\gamma(\pi) = \gamma(s_1)\gamma(s_2)\dots\gamma(s_n) \ .$$

Definition B.3 (Strategy)**.** *A* strategy *is a function $\sigma : \mathcal{O}^+ \to Act \cup \{done\}$ that maps a nonempty sequence of observations to a proposed action or the symbol $done \notin Act$, signaling that no further actions will be proposed.*

A path $\pi = s_0 s_1 \dots s_n$ *follows* a strategy σ if $s_i \xrightarrow{a_i, w_i} s_{i+1}$ for $a_i = \sigma(\gamma(s_0 s_1 \dots s_i))$ and $w_i \in \mathbb{Z}$, for all i, $0 \leq i < n$. A strategy σ is *terminating* if it does not generate any infinite path, in other words there is no infinite sequence where all its finite prefixes follow σ.

Given a subset of states $X \subseteq S(\mathcal{A})$ and a terminating strategy σ, the set of all maximal paths that follow the strategy σ in $\mathcal{A}$ and start from some state in X, denoted by $\sigma[X]$, is defined as follows:

$$\sigma[X] = \{\pi = s_0 s_1 \dots s_n \in \mathit{paths}(\mathcal{A}) \mid s_0 \in X, \pi \text{ follows } \sigma \text{ and } \sigma(\gamma(\pi)) = \mathit{done}\} \ .$$

The set of final states reached when following σ starting from X is defined as $\mathit{last}(\sigma[X]) = \{\mathit{last}(\pi) \mid \pi \in \sigma[X]\}$. Assuming a given observation function γ, we can now define a synchronizing strategy that will bring the system from any unknown initial state to the same single synchronizing state.

Definition B.4 (Synchronization)**.** *Given a WA $\mathcal{A}$, a strategy σ is* synchronizing *if σ is terminating and $|\mathit{last}(\sigma[S(\mathcal{A})])| = 1$. Further, $\mathcal{A}$ is* synchronizable *if it admits a synchronizing strategy.*

We limit our study to systems where we see no information about the current location and have a partial observability of the current weight so that we can distinguish whether its value is negative or nonnegative. This is the minimal possible observation as if we cannot observe anything about the weight then synchronization is impossible. Hence, we define the observation function γ to the set of observations $\mathcal{O} = \{<0, \geq 0\}$ by

$$\gamma((\ell, z)) = \begin{cases} <0 & \text{if } z < 0 \\ \geq 0 & \text{if } z \geq 0 \ . \end{cases}$$

We are interested in deciding whether a given WA is synchronizable under this observation function γ.

B.3 Polynomial Time Algorithm for Synchronizing

Let $\mathcal{A} = (L, Act, T, W)$ be a WA. We write $\ell \xrightarrow{a,w} \ell'$ for $\ell, \ell' \in L$ whenever $E(\ell, a) = \ell'$ and $w = W(\ell, a)$. A *cycle* in $\mathcal{A}$ starting in ℓ_0 is a path of the form $\ell_0 \xrightarrow{a_0,w_0} \ell_1 \xrightarrow{a_1,w_1} \dots \ell_n \xrightarrow{a_n,w_n} \ell_0$. The *weight* of the cycle is $\sum_{i=0}^{n} w_i$.

We now perform a series of checks in order to test whether we can synchronize from any possible initial state. The tests will give a necessary and sufficient condition for synchronization. At the end, we will argue that all the checks can be done in polynomial time. Therefore, we provide a polynomial time algorithm for deciding synchronizability. Furthermore, if all the checks succeed, we also construct a synchronizing strategy. It works in several phases, each concerning some of the tests. However, we note that although our algorithm decides synchronizability in polynomial time, the construction of a synchronizing strategy as an explicit function is not possible due to the fact that the lengths of the synchronizing sequences proposed by the strategy are unbounded (they depend on the initial weight values). We instead provide an algorithm, describing the unbounded strategy from any given initial state.

First, we check if the given $\mathcal{A}$, viewed as a labelled directed graph, has the following property.

Property 1. The graph $\mathcal{A}$ has a strongly connected component that is reachable from any location in $\mathcal{A}$.

If Property 1 is not satisfied, and such a bottom strongly connected component does not exist, clearly there is no synchronizing strategy for $\mathcal{A}$. From now on, assume that Property 1 holds. Taking advantage of this property, we define the first phase of the constructed synchronizing strategy σ.

▷ **Phase 1.** For every location ℓ let $home(\ell)$ be a sequence of actions that will bring ℓ into this strongly connected component and for any sequence of actions x let $\ell[x]$ be the location that we will reach from ℓ after performing the sequence x (note that this is well defined due to the fact that $\mathcal{A}$ is deterministic). Our synchronizing strategy will start by performing the action sequence $x_1x_2x_3 \dots x_n$ where $x_1 = home(\ell_1)$, $x_2 = home(\ell_2[x_1])$, $x_3 = home(\ell_3[x_1x_2])$, $\dots, x_n = home(\ell_n[x_1x_2 \dots x_{n-1}])$ assuming that $L = \{\ell_1, \ell_2, \dots, \ell_n\}$. We shall refer to this technique as *sequentialization*: intuitively, even if the initial location is unknown, we can perform given actions for each possible initial location, meanwhile tracking where we move if the actual initial location was different, and execute these steps in a sequence for each possible location. ◁

Example B.5. *We illustrate Phase 1 on the system below in Figure B.2. We first execute $home(\ell_1) = a$. Meanwhile both ℓ_2 and ℓ_3 move to ℓ_2. Therefore, we proceed with $home(\ell_2) = ba$. Therefore, if we started in ℓ_3 we are now in ℓ_4, too. Since ℓ_4 is in a bottom strongly connected component, $home(\ell_4)$ is the empty sequence and we are done.*

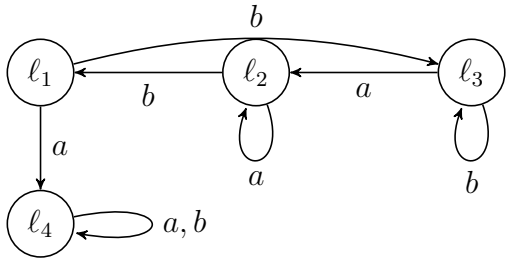


Figure B.2: Example of sequentialization (the word *aba* will bring all locations to ℓ_4)

Consequently, after Phase 1, we are for sure in the strongly connected component. Within this component, we check the following property.

Property 2. Let $\mathcal{A}$ be strongly connected. In $\mathcal{A}$, there is a cycle with weight 1 and a cycle with weight -1.

Lemma B.6. *If Property 2 is not satisfied then there is no synchronizing strategy.*

Proof. Assume that $\mathcal{A}$ is synchronizable. Then there must be a positive and a negative cycle in $\mathcal{A}$. Further, for any location ℓ, the states $(\ell, 0)$ and $(\ell, 1)$ can be synchronized. Therefore, there is a path from both these states to some state (ℓ', z) for some $z \in \mathbb{Z}$. Since $\mathcal{A}$ is strongly connected, there is also a path from (ℓ', z) back to the state (ℓ, z') for some $z' \in \mathbb{Z}$. Consequently, there are two cycles from ℓ with weights z' and $z' - 1$, respectively; moreover, these weights can be chosen non-zero due to existence of a positive cycle. Hence the weights of these two cycles are relative primes and in combination with the presence of a positive and negative cycle in $\mathcal{A}$, this implies the existence of cycles with the weights $+1$ and -1. □

From now on, assume that $\mathcal{A}$ is strongly connected and Property 2 holds. Observe that, consequently, there are $+1$ and -1 cycles starting and ending in each location ℓ. Let us denote the corresponding sequences of actions by ℓ^+ and ℓ^-. The first, and rather naive, use of these cycles is to get the weight component of the state close to zero.

▷ **Phase 2.** We extend our strategy σ by performing the ± 1 cycles until we see a change in our observation. Assuming we start with nonnegative observation, Phase 2 ends at the moment when a negative observation is reached (and symmetrically for the other case). To this end, assuming $L = \{\ell_1, \ldots, \ell_n\}$, we employ the sequentialization technique again. We first execute the word ℓ_1^- for the -1 cycle from ℓ_1 and keep track of the resulting locations $\{\ell'_1, \ldots, \ell'_n\}$. Note that their weights could have increased instead, say by at most c_1. Next we execute ${\ell'_2}^-$ exactly $(c_1 + 1)$-times, so that even if the initial location was ℓ_2, after this many cycles the weight decreased no matter how it increased by performing ℓ_1^-. Meanwhile ℓ'_3 changes to ℓ''_3 and its weight could have in total increased by at most c_2. We thus execute ${\ell''_3}^-$ exactly $(c_2 + 1)$-times and so on for all locations cyclically (starting again at the first location once we went through all of them) until the weight decreases below zero. This process terminates since whenever performing cycles for a particular location, its weight (if we indeed started in the respective location) drops below any previous value. ◁

Example B.7. *We illustrate Phase 2 on the system below in Figure B.3 when the observation is nonnegative. We first execute $\ell^- = a$. Meanwhile r loops under a and increases by $c_1 = 3$. Therefore, we proceed with repeating $r^- = bab$ for 4 times. This in turn makes ℓ return again to ℓ with value increased by $c_2 = 4 \cdot 3 = 12$. Next we repeat ℓ^- for 13 times etc.*

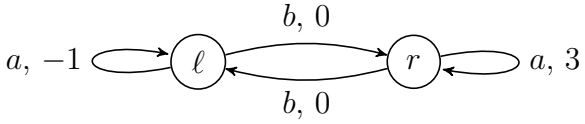


Figure B.3: Example illustrating Phase 2

We are now guaranteed that right after Phase 2, the observation has just changed. Therefore, we are now in a state (ℓ, z) for some $\ell \in L$ and

$$0 \leq z < M \text{ if } \gamma((\ell, z)) = \geq 0 \quad \text{or} \quad -M \leq z < 0 \text{ if } \gamma((\ell, z)) = < 0$$

where M is the largest absolute value of any weight used in $\mathcal{A}$. Therefore, there are finitely many states we can be at. Now that we have a bound how far we are from zero, we can make a better use of ± 1 cycles and derive for each location, where we possibly might be at, its weight.

▷ **Phase 3.** Once again, we employ sequentialization. Assuming first that we are in location $\ell_1 \in L$ of a state (ℓ_1, z) with $-M \leq z < M$, we can perform

a sequence of actions corresponding to -1 or $+1$ cycle from ℓ_1 (depending on whether $\gamma((\ell_1, z)) = \geq 0$ or $\gamma((\ell_1, z)) = <0$) until the observation changes at the end of the cycle when we are again in ℓ_1. If we indeed started in the location ℓ_1, we know that we are now in the state $(\ell_1, -1)$ if $\gamma((\ell_1, z)) = \geq 0$, or $(\ell_1, 0)$ if $\gamma((\ell_1, z)) = <0$. If the weight in the reached states did not change from nonnegative to negative (or the other way round) even after performing M cycles, we know for sure that we were not in the location ℓ_1. The situation where we started in a different location than ℓ_1 and the weight observation still changed as expected simply adds an extra (false) hypothesis that can be eliminated (as shown later on in this section).

Now consider what would happen, until now, if we were instead in location $\ell_2 \in L$ at the moment before we started to perform the -1 or $+1$ cycles for ℓ_1. After playing according to the strategy above, we would be now in a possibly different location ℓ_2' with weight in the range $[-M', M']$ where M' can be computed from M and the strategy performed so far. We can now start performing -1 or $+1$ cycle from ℓ_2' exactly as before in order to determine the exact weight in this location (provided we started in ℓ_2) and we continue like this with handling ℓ_3 etc., for each location in L. ◁

Example B.8. *We illustrate Phase 3 on the system below in Figure B.4. If the observation is nonnegative, the current weight is at most* 2 *and we start with repeating bbaa, a* -1 *cycle for* ℓ*, for at most* $M = 3$ *times. If the observation remains nonnegative after this sequence (case 1) we must be in* r*. Otherwise (case 2), we stop when the observation changes and if we are in* ℓ *the current weight is* -1*. Meanwhile* r *returned back to* r *and could have increased its weight to at most* 5 *in case 2. Then we proceed with repeating bbaa at most* 6 *times to get for sure to* $(r, -1)$*. In case 1, the observation is negative and the weight is at least* -2*. Hence, we repeat aab, a* $+1$ *cycle for* r*. Say that the observation changes after two repetitions. Then we are either in* $(r, 0)$ *or in the meanwhile achieved* $(\ell, -3)$*. (The latter is, however, impossible here since the observation would remain negative.)*

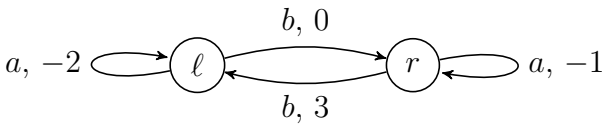


Figure B.4: Example illustrating Phase 3

We conclude that after Phase 3 we must be in one of the states from the set

$$\{(\ell_1, z_1), (\ell_2, z_2), \ldots, (\ell_n, z_n)\}$$

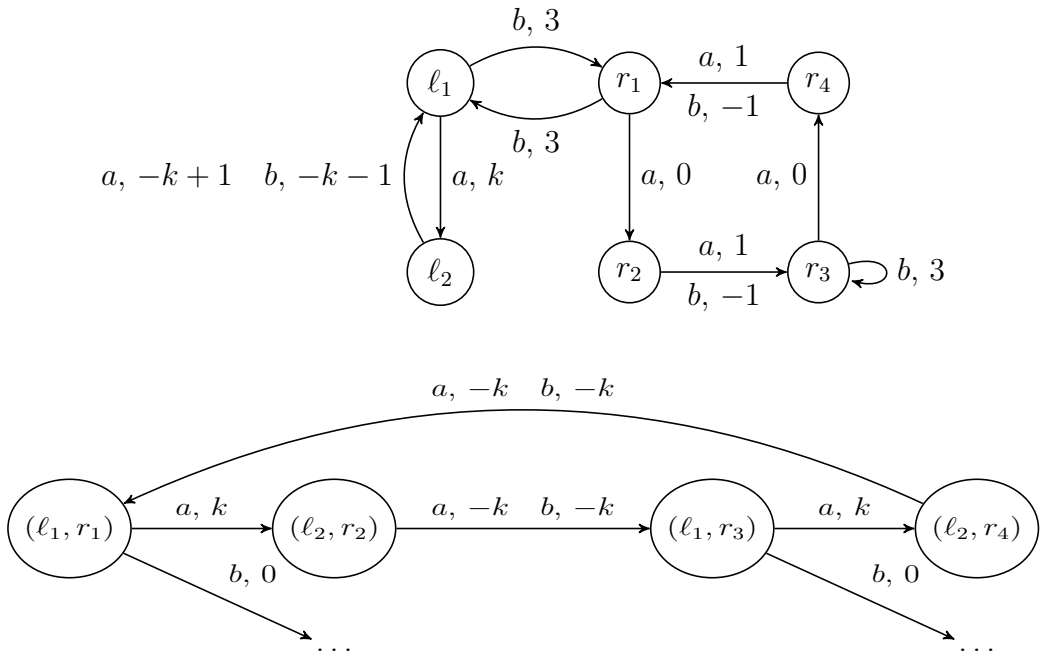


Figure B.5: Example of difference graph

called the *hypothesis set*, where all z_i's are exactly known. We can w.l.o.g. assume that all locations in the assumption are pairwise different. Indeed, we can perform a number of ± 1 cycles from the location that appears in the hypothesis set more times and determine which one of the weights is still feasible (at most one is). Note that the size of the hypothesis set is thus at most $|L|$.

The next task is to distinguish between these hypotheses. For each pair of locations, assuming their weights from the hypothesis set, there must be a way to synchronize them. We present three tests such that at least one of them must be passed by each pair. All tests refer to the following notion of difference graph.

Definition B.9 (Difference Graph)**.** *The* difference graph *of a WA $\mathcal{A}$ is a weighted graph $G^{\mathcal{A}} = (V, E)$ with $E \subseteq V \times \mathbb{Z} \times V$ such that $V = L \times L$, and for every $a \in Act$ we have $\Big((\ell, \ell'),\, W(\ell, a) - W(\ell', a), (E(\ell, a), E(\ell', a))\Big) \in E$.*

In other words, $G^{\mathcal{A}}$ is a synchronous product of two $\mathcal{A}$'s, where each edge weight is the difference of edge weights in the first and the second component.

Example B.10. *Consider the system on the upper part of Figure B.5, parametrized by $k \in \mathbb{Z}$. We depict a part of its difference graph on the lower part of the figure.*

We have already seen how to distinguish states with the same location using ± 1 cycles. For all pairs of locations (ℓ_1, ℓ_2) where $\ell_1 \neq \ell_2$, we run the following three tests.

Property 3. There is a path in $G^{\mathcal{A}}$ from (ℓ_1, ℓ_2) to (ℓ, ℓ) for some $\ell \in L$.

Property 4. There is a path in $G^{\mathcal{A}}$ from (ℓ_1, ℓ_2) to (ℓ_1', ℓ_2') such that there is a cycle of nonzero weight (positive or negative) in $G^{\mathcal{A}}$ starting in (ℓ_1', ℓ_2').

In order to define the last Property 5, we need additional notions and reasoning. If Property 4 is not satisfied for a given pair (ℓ_1, ℓ_2) then every cycle in $G^{\mathcal{A}}$ reachable from (ℓ_1, ℓ_2) has zero weight. Therefore, whenever any cycle C in $G^{\mathcal{A}}$ is performed in $\mathcal{A}$ starting from location ℓ_1 or ℓ_2, the weight changes in both cases by the same value, called the *projected weight* $\mathcal{P}(C)$ of C. Although $G^{\mathcal{A}}$ may be disconnected, we may w.l.o.g. assume that (ℓ_1, ℓ_2) is a node in $G^{\mathcal{A}}$ that is a part of some strongly connected component (otherwise, we bring it there, using sequentialization).

Lemma B.11. *For every strongly connected component S of $G^{\mathcal{A}}$ satisfying Property 2 and not satisfying Property 4, there is a number p such that $1 \leq p \leq |L|$ and for any node (pair of locations) in S there are cycles C^+, C^- starting in this node with $\mathcal{P}(C^+) = p$ and $\mathcal{P}(C^-) = -p$. Moreover, such p can be computed in polynomial time.*

Proof. Let (ℓ_1, ℓ_2) be an arbitrary node in S. Due to Property 2, there is a cycle from ℓ_1 in $\mathcal{A}$ with weight $+1$. We repeatedly perform this $+1$ cycle and follow the same behaviour from (ℓ_1, ℓ_2) in $G^{\mathcal{A}}$. At the end of each cycle, the first component in $G^{\mathcal{A}}$ will be in the location ℓ_1 and the second component in one of the $|L|$ possible locations. By the pigeon-hole principle, after performing the $+1$ cycle at most $|L|$ times, we will find a repeated pair in $G^{\mathcal{A}}$. Hence we found a cycle C^+ in $G^{\mathcal{A}}$ with zero weight in $G^{\mathcal{A}}$, due to the violation of Property 4, and with the projected weight $0 < \mathcal{P}(C^+) \leq |L|$. By the same arguments, but using the fact about the existence of -1 cycle in $\mathcal{A}$, we can find a cycle C^- in $G^{\mathcal{A}}$ with the projected weight $0 > \mathcal{P}(C^-) \geq -|L|$.

Let us now argue that for each node we can choose cycles with absolute weights equal to a fixed integer. Let $p_{\min} := \min\{|\mathcal{P}(C)| \mid C \text{ is a cycle in } S\}$ denote the smallest projection over all cycles in the strongly connected component S. We claim that for any pair of states (ℓ_1, ℓ_2) in S, there are cycles in S starting in (ℓ_1, ℓ_2) with projected weights $p_{\min}$ and $-p_{\min}$. Indeed, note that there is a cycle C in S with $|\mathcal{P}(C)| = p_{\min}$ and there are cycles D^+ and D^- from (ℓ, ℓ') that visit some state of C and have positive and negative projected weight, respectively.

Now by repeating C on the way either in D^+ or in D^-, we construct cycles E^+ and E^- with $0 < \mathcal{P}(E^+) \leq p_{\min}$ and $0 > \mathcal{P}(E^-) \geq -p_{\min}$, respectively. By minimality of $p_{\min}$, we obtain $\mathcal{P}(E^+) = p_{\min}$ and $\mathcal{P}(E^-) = -p_{\min}$. Note that, moreover, for each cycle C is S, $\mathcal{P}(C)$ is a multiple of $p_{\min}$. And vice versa, for each multiple of $p_{\min}$, there is a cycle with such projected weight in any node of S. Therefore, we can perform the pigeon-hole construction of a cycle (in polynomial time), obtaining a weight $0 < p \leq |L|$, and we are guaranteed that from each node there are cycles with projected weights p and $-p$, respectively (although p is not necessarily minimal). □

Consequently, for each strongly connected component S, we have $0 < p \leq |L|$. For S, we define a reachability problem on a graph $\mathcal{A}_S = (V, \to)$ where

- $V = (L \times \{0, \ldots, p-1\}) \times (L \times \{0, \ldots, p-1\}) \cup \{\mathit{separated}\}$, and
- for each $v = \Big((\ell_1, z_1), (\ell_2, z_2)\Big)$ and $a \in \mathit{Act}$, let $v' = \Big((\ell_1', z_1'), (\ell_2', z_2')\Big)$ where $\ell_1' = E(\ell_1, a)$, $\ell_2' = E(\ell_2, a)$ and $z_1' = z_1 + W(\ell_1, a) + \alpha \cdot p$ and $z_2' = z_2 + W(\ell_2, a) + \alpha \cdot p$ for the unique $\alpha \in \mathbb{Z}$ such that the larger of z_1', z_2' lies in the interval $[0, p-1]$; we set
  - $v \to v'$ if $v' \in V$, i.e. the lower weight is also nonnegative, and
  - $v \to \mathit{separated}$, otherwise, i.e. the lower weight is negative.

We say that the graph $\mathcal{A}_S$ is *distinguishing* for a pair of locations $(\ell_1, \ell_2) \in S$ if from any initial node $\Big((\ell_1, z_1), (\ell_2, z_2)\Big)$, for each $0 \leq z_1, z_2 \leq p-1$, we can reach the node *separated*. Note that the size of $\mathcal{A}_S$ is at most $|L|^4 + 1$, hence polynomial in $|\mathcal{A}|$. Now we state the final test.

Property 5. If (ℓ_1, ℓ_2) belongs to a strongly connected component S of $G^{\mathcal{A}}$ then the graph $\mathcal{A}_S$ is distinguishing for (ℓ_1, ℓ_2).

Example B.12. *Consider the difference graph of Figure B.5. Observe that there is no path from (ℓ_1, r_1) to a pair with identical components, as well as no nonzero cycle. The length p is equal 2 here. If $k \geq 2$ then we have $((\ell_1, 0), (r_1, 0)) \to$ separated as action a immediately creates a large enough difference. If $k = 1$, then separated is still reachable from $((\ell_1, 0), (r_1, 0))$, but only after aa is taken. Then both weights are 1 and the next action a creates the distinguishing difference as the weights would now be $2, 1$, i.e. transformed to $0, -1$.*

Supposing each pair of locations satisfies Property 3 or Property 4 or Property 5, we can iteratively decrease the size of the hypothesis set until it becomes a singleton as shown in the next phase.

▷ **Phase 4.** We employ sequentialization again. We pick any two states from the current hypothesis set and eliminate at least one of them as described below. Meanwhile, we update all remaining states from the hypothesis set to their current states. We repeat this procedure until the hypothesis set becomes a singleton. Let (ℓ_1, z_1) and (ℓ_2, z_2) be the currently explored pair from the hypothesis set.

First, if Property 3 holds we perform the sequence of actions that brings both locations into a single location. Afterwards, if their respective weights are different, using the ± 1 cycles, we detect at least one of the weights impossible as above. Thus we decrease the size of the hypothesis set.

Second, if Property 4 holds then we can extend our strategy σ by executing the sequence of actions that brings (ℓ_1, ℓ_2) to some (ℓ_1', ℓ_2') where we can repeatedly execute actions on the nonzero cycle in $G^{\mathcal{A}}$ until the weights in the pair of states reached after this sequence are sufficiently (see below) far away from each other. Assume w.l.o.g. that the weights z_1', z_2' of the two reached states are both positive and $z_1' < z_2'$ (the other situations are symmetric). Now from the location with the lower weight (ℓ_1'), we enter a simple cycle in $\mathcal{A}$ with the minimal (negative) weight and start executing it. This ensures that if we started from ℓ_1 or ℓ_2, then the observation will change to negative in n_1 or n_2 steps, respectively, where $n_1 < n_2$ (since $|z_2' - z_1'|$ was sufficiently large) and we can compute these numbers. If the observation changes after exactly n_1 steps, we eliminate the state corresponding to ℓ_2 from the hypothesis set. If the observation changes after exactly n_2 steps, we eliminate the state corresponding to ℓ_1 from the hypothesis set. If the observation changes after a different number of steps, we eliminate both.

Third, let Property 5 hold (and Property 4 not) and (ℓ_1, ℓ_2) be in a strongly connected component S of $G^{\mathcal{A}}$. By Lemma B.11, we have zero cycles in $\mathcal{A}_S$ with projected weights p and $-p$ where $0 < p \leq |L|$. We perform these cycles until the larger weight is in $[0, p-1]$. If the lower weight is negative at this moment, the current observation eliminates one of the hypotheses. Otherwise, the weights in both states are in $[0, p-1]$. Due to Property 5 we have a strategy to reach *separated* in $\mathcal{A}_S$, inducing a strategy in $\mathcal{A}$ by inserting the $-p$ and p cycles. Upon reaching *separated* in $\mathcal{A}_S$, the observation in $\mathcal{A}$ proves one of the two hypotheses impossible. (If at any moment throughout the process, an unexpected change of observation occurs, we eliminate the respective hypothesis from the set immediately.)

Once the hypothesis set is a singleton, we know precisely the current state. Finally, we deterministically reach a fixed location and fixed weight (by performing ± 1 cycles) and thus synchronize. ◁

The stated properties are not only sufficient, but also necessary conditions for synchronizability:

Lemma B.13. *Let $\mathcal{A}$ be a strongly connected WA satisfying Property 2. Then $\mathcal{A}$ is synchronizable if and only if for each pair of locations (ℓ_1, ℓ_2) either Property 3 or Property 4 or Property 5 is satisfied.*

Proof. The "if"-part follows from the previously constructed synchronizing strategy. For the "only-if"-part, assume that there is a pair (ℓ_1, ℓ_2) satisfying neither Property 3, nor Property 4, nor Property 5. By the last one, there are weights $z_1, z_2 \in [0, p-1]$ such that the node *separated* is not reachable from the configuration $\mathit{init} = \big((\ell_1, z_1), (\ell_2, z_2)\big)$ in the graph $\mathcal{A}_p$. For a contradiction, assume that $\mathcal{A}$ admits a synchronizing strategy σ. When σ is applied to initial states (ℓ_1, z_1) and (ℓ_2, z_2), we obtain two paths π_1 and π_2, inducing two sequences of observations $\gamma(\pi_1)$ and $\gamma(\pi_2)$. Comparing the respective elements in the two sequences, there are two cases.

In the first case, observations will never differ. Since σ is synchronizing, it brings both states (ℓ_1, z_1) and (ℓ_2, z_2) eventually into the same state, in particular to the same location, witnessing Property 3 and contradicting to our assumption.

In the second case, after a certain number of steps, the observations of the current states (ℓ_1', z_1') and (ℓ_2', z_2') of the two path will differ, w.l.o.g. $z_1' < 0 \leq z_2'$. Since Property 4 is not satisfied, by Lemma B.11 there are cycles increasing and decreasing weight in both ℓ_1 and ℓ_2 by p. The two paths π_1, π_2 produced by the strategy σ in $\mathcal{A}$ induce two sequences $\hat{\pi}_1, \hat{\pi}_2$ where the ith elements are both increased/decreased by $\alpha_i \cdot p$ for some $\alpha_i \in \mathbb{Z}$ so that the larger one is in $[0, p-1]$. These sequences straightforwardly induce a path in $\mathcal{A}_S$, where S is the strongly connected component of *init*. Since *separated* cannot be reached from *init*, the smaller weight is always in $[0, p-1]$, too. Let $\hat{z}_1', \hat{z}_2'$ denote the weights in $\mathcal{A}_S$ when σ achieves z_1', z_2'. Since $z_2' \geq 0$, $\hat{z}_2' < p$, and $\hat{z}_2' \equiv z_2' \pmod{p}$, we obtain $\hat{z}_2' \leq z_2'$. Therefore, by $\hat{z}_2' - \hat{z}_1' = z_2' - z_1'$ we also get $\hat{z}_1' \leq z_1'$. Since $z_1' < 0$, we obtain $\hat{z}_1' < 0$, a contradiction. □

B.4 Complexity

We can now state our main theorem:

Theorem B.14. *The synchronizability problem for deterministic weighted automata is decidable in polynomial time.*

Proof. Properties 1-5 form sufficient and necessary conditions for the existence of a synchronizing strategy for $\mathcal{A}$ by Lemma B.6 and Lemma B.13. Moreover, all properties can be verified in polynomial time. Indeed, the size of $G^{\mathcal{A}}$ is polynomial in $|\mathcal{A}|$ and p necessary for constructing $\mathcal{A}_S$ is computable in polynomial time by Lemma B.6, and the presence of ± 1 cycles is decided in polynomial time by Theorem B.17 as discussed in the rest of this section. □

We now prove that the presence of ± 1 cycles can be decided in polynomial time. We assume a weighted graph $G = (V, E)$ where V is a finite set of nodes and $E \subseteq V \times \mathbb{Z} \times V$ are the edges written as $u \xrightarrow{w} v$ whenever $(u, w, v) \in E$. A *path* in G is a sequence of edges $v_0 \xrightarrow{w_0} v_1 \xrightarrow{w_1} \dots \xrightarrow{w_{n-1}} v_n$. A *weight of a path* π is defined as $|\pi| = \sum_{i=0}^{n-1} w_i$. A *$k$-cycle* is a path π where $v_0 = v_n$ such that $k = |\pi|$.

Remark B.15. *We first briefly discuss related problems and point to severe differences, preventing us from adapting the existing results. On the one hand, we note that the problem whether there is a k-cycle, where k is a part of the input, is NP-hard (see Section B.6). On the other hand, it is a classical result [87] that existence of 0-cycles is decidable in polynomial time. The result can be proven by a reduction to linear programming. The idea is the following. For each transition, there is a variable encoding the frequency of the transition on the desired cycle. Encoding of Kirchhoff's flow-preservation laws then ensures that the frequencies indeed induce a cycle. Finally, the sum of transition weights multiplied by the frequencies is required to be 0. From every rational solution, we can by multiplication obtain an integer solution, and thus a realizable cycle. Since 0 multiplied by any number remains 0, we thus obtain a 0-cycle. In contrast, in our setting, this idea cannot be used. Indeed, suppose we require the frequency-weighted sum of edge-weights to be 1. Since the frequencies and thus also the number 1 must be multiplied by an a priori unknown integer, in order to obtain an integer solution, the resulting total weight is not 1. Asking instead directly for an integer solution to the system is an instance of integer linear programming, which is an NP-hard problem. Instead of using linear programming, we employ a number theoretic arguments and exploit Dijkstra's shortest path algorithm on graphs where weights are counted modulo various numbers. Finally, note that although we can decide the existence of ± 1-cycles in polynomial time, the length (number of edges) of the shortest one may still be exponential. For instance, consider a single vertex with two self-loops labelled by $2^n + 1$ and -2.* △

The discussion suggests that number theoretic techniques have to be applied. We reduce our problem to the problem whether the greatest common divisor

of all cycles in a graph is 1. Formally, for a weighted graph G, let the *period* $\gcd(G)$ denote $\gcd\{k \mid k \in \mathbb{Z}, G \text{ has a } k\text{-cycle}\}$.

Proposition B.16. *For every strongly connected weighted graph G, there is a 1-cycle and a -1-cycle in G if and only if there is a positive and a negative cycle in G and $\gcd(G) = 1$.*

Proof. The 'Only-if' direction is trivial. For the 'If' direction, $\gcd(G) = 1$ yields by Bézout's identity an equality

$$1 = \alpha_1 \cdot k_1 + \cdots + \alpha_n \cdot k_m \tag{B.1}$$

for some $m \in \mathbb{N}$, $\alpha_i \in \mathbb{Z}$, and k_i being the weight of some cycle c_i in G, and where, moreover, some $k_p > 0$ and some $k_n < 0$. Note that these numbers can be extracted using the extended Euclidean algorithm. First, we argue, we can choose all $\alpha_i \geq 0$ so that Equation (B.1) still holds.

Whenever $\alpha_i < 0$ with k_i positive, we increase α_i by $x \cdot (-k_n)$ for some $x \in \mathbb{N}$ so that it becomes positive. Further, we increase α_n by $x \cdot k_i$, thus preserving Equation (B.1). For negative k_i, we proceed similarly, using k_p and α_p instead. Since this procedure only increases α's, they all eventually become positive.

Nonnegative coefficients α_i determine the number of repetitions of each cycle c_i. Since these cycles may be disconnected, this does not yield a single 1-cycle yet. To this end, we consider a negative cycle visiting each vertex of G, guaranteed by assumptions. Let $-\omega$ denote its weight. We construct a 1-cycle by executing this cycle and on the way, whenever reaching a vertex where the cycle c_i originates, we execute c_i for $(\omega + 1) \cdot \alpha_i$ times. A -1-cycle is constructed similarly. □

Theorem B.17. *The presence of both a 1-cycle and at the same time a -1-cycle in a weighted graph G is decidable in polynomial time. Moreover, such cycles can be effectively constructed.*

Proof. Deciding presence of a negative cycle and producing a witness can be done in polynomial time using, for instance, Bellman-Ford algorithm (see e.g. [8]); the same holds for positive cycles by swapping the signs.

The period of a graph can be computed in polynomial time, too. Indeed, the result for unweighted graphs (all weights are one) was proven in [84]. Further, [4] suggests an extension of the technique to weighted graphs. Since [84] is to the best of our knowledge not accessible electronically (the only hardcopy of the report is located at library of Stanford University) and the correctness of the extension to weighted graphs is not proven in [4], we also provide our own proof, using supposedly different techniques. Section B.5 gives the details. □

Remark B.18. *The polynomial time algorithm for deciding synchronizability is relying only a single observation, testing whether the accumulated weight is negative or nonnegative. In a more general setting, we may consider a richer set of observations checking whether the weights are less-than/greater-or-equal to a given number of integer values. The techniques in this paper can be directly reused to handle this more generation situation and the only check that must be modified is Property 5. Here, if some observations are far away from each other (the integers that they test have distance more than p) then it is sufficient to check if at least of them succeeds, otherwise the graph $\mathcal{A}_S$ is extended to include weights in the range $[0, kp-1]$ where k is the number of observations that are close to each other so that all of them are considered in the check for distinguishability of a given pair of locations. As the observations are part of the input (of the problem description), this still creates a graph with only polynomially many nodes.*

B.5 Algorithm for Finding Period $\gcd(G)$ of Graph G

We show how to compute the period (gcd of all cycles) of a weighted graph. Although it is actually sufficient to examine simple cycles in the graph, there are still exponentially many of them. Hence, we compute $\gcd(G)$ in a more efficient way by Algorithm B.1.

The idea is to pick any cycle, say with weight ω (for simplicity of notation, assume it is positive). If we knew the primal decomposition $\omega = p_1^{k_1} \cdots p_n^{k_n}$, we could search, for each p_i, for a cycle with weight not divisible by p_i. If we find such a cycle for each i, then $\gcd(G) = 1$. However, prime decomposition is not known to be computable in polynomial time. Therefore, we look instead for a cycle with an arbitrary weight $\bar{\omega}$ not divisible by ω, i.e. $\bar{\omega} \not\equiv 0 \pmod{\omega}$. We use $\gcd(\omega, \bar{\omega})$ as the new ω and iterate this procedure until all cycles have weight divisible by the current ω. Intuitively, $\bar{\omega}$ eliminates at least one factor from the primal decomposition, although we cannot upfront determine which one.

The invariant of Algorithm B.1 is that $\gcd(G)$ divides ω, following from $\gcd(G) = \gcd(\gcd(G), \gcd(\omega, \bar{\omega}))$ for any $\bar{\omega}$-cycle. Further, whenever $\omega \neq \gcd(G)$, there is an $\bar{\omega}$-cycle with $\gcd(\bar{\omega}, \omega) \neq \omega$, which is thus found in line 2, implying partial correctness of the algorithm. Termination follows from the unique finite primal decomposition by the fundamental theorem of arithmetic.

Algorithm B.1 Computation of $\gcd(G)$ for a weighted graph G

Input: Weighted graph $G = (V, E)$
Output: $\gcd(G)$
1: Pick an arbitrary simple cycle in G with positive weight, denoted by ω
2: **while** there is $\bar{\omega}$-cycle with $\bar{\omega} \not\equiv 0 \pmod{\omega}$ **do** ▷ $\gcd(\bar{\omega}, \omega) \neq \omega$
3: $\quad \omega \leftarrow \gcd(\omega, \bar{\omega})$ ▷ using e.g. Euclidean algorithm
4: **return** ω

We now implement the test in line 2, returning the value $\bar{\omega}$, by Algorithm B.2. Instead of finding an $\bar{\omega}$-cycle with $\bar{\omega} \not\equiv 0 \pmod{\omega}$, we decompose the cycle into an edge and the remaining path that form the cycle.

Algorithm B.2 Detection of a cycle with weight not divisible by ω

Input: Weighted graph $G = (V, E)$, a number $\omega \in \mathbb{N}$
Output: Weight $\bar{\omega}$ of some cycle in G with $\bar{\omega} \not\equiv 0 \pmod{\omega}$, ff if there is none
1: **for all** $u \xrightarrow{x} v$ in G **do**
2: $\quad$ **if** there is a y-path from v to u with $y \not\equiv -x \pmod{\omega}$ **then**
3: $\quad\quad$ **return** $x + y$
4: **return** ff

This small trick allows us to use a modification of Dijkstra's algorithm for shortest paths. This modification is inspired by [71], where a similar idea was used in the context of minimum cycle bases.

Definition B.19. *For $d \in \mathbb{Z}$ and $p \in \mathbb{N}$, a path π is a (d, p)-path if $|\pi| \not\equiv d \pmod{p}$.*

In line 2 of Algorithm B.2, we ask for the existence of a $(-x, \omega)$-path. We find more convenient to be more specific and, in Algorithm B.3, we compute in some sense the shortest (d, p)-path. In order to avoid problems with negative weights in the context of shortest paths, we change every edge $u \xrightarrow{w} v$ in G into $u \xrightarrow{w'} v$ where $w' = w \bmod p$, i.e. we modify its weight so that the new weight w' belongs to the interval $[0, p)$ and satisfies $w' \equiv w \pmod{p}$. Clearly, the set of (d, p)-paths stays the same under this transformation. We denote the resulting weighted graph by $G^{\bmod p}$.

Given vertices *start* and *target*, Algorithm B.3 first computes the shortest paths π_v from *start* to every node v in $G^{\bmod p}$. If the shortest path to *target* happens to be a (d, p)-path, we are done. Otherwise, for every vertex v, we compute the *shortest non-Dijkstra path*, i.e. the shortest path to v with a remainder modulo p different from that of $|\pi_v|$. Technically, in a Dijkstra-like computation, key_v stores the weight of the current shortest non-Dijkstra path

Algorithm B.3 Dijkstra's algorithm modified for the shortest (d, p)-path

Input: Weighted graph $G = (V, E)$, $start, target \in V$, $d \in \mathbb{Z}$, $p \in \mathbb{N}$
Output: Weight of a shortest path π in $G^{\bmod p}$ from *start* to *target* with $|\pi| \not\equiv d \pmod p$, ff if there is none

1: $G \leftarrow G^{\bmod p}$ ▷ constructing $G^{\bmod p}$ with only positive weights
2: for each $v \in V$, compute the shortest path π_v from *start* to v ▷ using e.g. Dijkstra's algorithm
3: **if** $|\pi_{target}| \not\equiv d \pmod p$ **then**
4: **return** $|\pi_{target}|$
5: **else**
6: **for all** $v \in V$ **do**
7: $key_v \leftarrow \min\{|\pi_u| + w \mid u \xrightarrow{w} v,\ |\pi_u| + w \not\equiv |\pi_v| \pmod p\}$ ▷ $\min \emptyset = \infty$
8: enqueue all $v \in V$ with $key_v < \infty$ to the priority queue Q ordered by *key*
9: **while** $Q \neq \emptyset$ **do**
10: $u \leftarrow$ dequeue from Q
11: **for all** $u \xrightarrow{w} v$ **do**
12: $k \leftarrow key_u + w$
13: **if** $k < key_v$ and $k \not\equiv |\pi_v| \pmod p$ **then**
14: $key_v \leftarrow k$
15: enqueue v to Q
16: **if** $key_{target} = \infty$ **then return** ff
17: **else return** key_{target}

to v, i.e. $key_v \not\equiv |\pi_v| \pmod p$. This ensures that we always have an alternative remainder to the one given by $|\pi_v|$. Consequently, either $|\pi_v|$ or key_v will always have a remainder modulo p different from d. Formally, correctness of Algorithm B.3 is a consequence of the following lemma:

Lemma B.20. *Every shortest non-Dijkstra path from u to v is*

a) either of the form $\pi_s \xrightarrow{w} v$, where π_s is a shortest path from u to s,

b) or $\rho_s \xrightarrow{w} v$ where ρ_s is a shortest non-Dijkstra path from u to s.

Proof. Suppose $\sigma \xrightarrow{w} v$ is a shortest non-Dijkstra path from u to v and the condition a) does not hold, i.e. $|\sigma| > |\pi_s|$ where $s = last(\sigma)$. We want to prove that then condition b) must hold. Since $\pi_s \xrightarrow{w} v$ is shorter than a shortest non-Dijkstra path $\sigma \xrightarrow{w} v$, it cannot be a non-Dijkstra path and hence $|\pi_s| + w \equiv |\pi_v| \pmod p$. Therefore, for any non-Dijkstra path σ' from u to s, i.e. with $|\sigma'| \not\equiv |\pi_s| \pmod p$, we thus obtain $|\sigma'| + w \not\equiv |\pi_v| \pmod p$, implying $\sigma' \xrightarrow{w} v$ is a non-Dijkstra path. Consequently, the no n-Dijkstra path σ must be a shortest one. □

In summary, Algorithm B.2 decides in polynomial time for a given graph G and a weight ω whether G has a $\bar{\omega}$-cycle such that $\bar{\omega} \not\equiv 0 \pmod{\omega}$. The use of Algorithm B.3 as a subroutine for line 2 is correct due to the fact that (i) every cycle in G has the same value in $G^{\bmod \omega}$ modulo ω and (ii) whenever there is a (d, p)-path between two nodes, there is also a shortest one.

B.6 Detecting k-Cycles in Weighted Graphs is NP-Hard

We want to show that given a weighted graph G and an integer k, the existence of a k-cycle in G is NP-hard. The folklore reduction from the subset sum problem [118] to the existence of a path of a given weight k in a weighted graph must be slighly modified in order to work also for cycles.

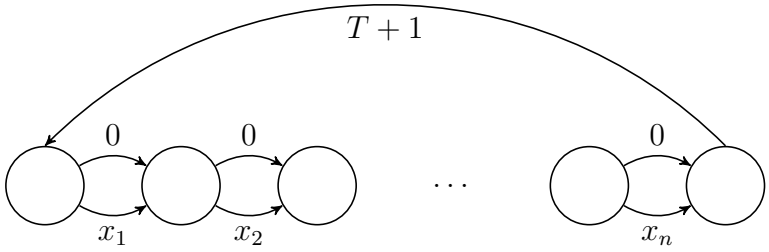


Figure B.6: Subset sum is solvable if and only if there is a $(2T+1)$-cycle.

Lemma B.21. *Given a weighted graph G and an integer k, the problem whether G contains a k-cycle is NP-hard.*

Proof. We provide a reduction from the subset sum problem. Given an instance of subset sum $(\{x_1, x_2, \ldots, x_n\}, T)$, where $T \in \mathbb{N}$ and $x_i \in \mathbb{N}$ for all i, $1 \leq i \leq n$, the subset sum question is whether there is a subset $X \subseteq \{x_1, x_2, \ldots, x_n\}$ such that $\sum X = T$. From the subset sum instance, we construct a weighted graph G as in Fig. B.6. Clearly, the subset sum instance has a solution if and only if there is a $(2T+1)$-cycle in G. Notice that the extra edge with weight $T+1$ is necessary as it allows to take the cycle only once in order to compose the remaining number T from the edges with weights x_i, $1 \leq i \leq n$. Should the cycle be taken more than once, its weight will for sure be at least $2T+2$ and hence it will not be a $(2T+1)$-cycle. □

B.7 Conclusion

We have shown that the synchronization problem for deterministic WA under (minimal) partial observability is decidable in polynomial time. This result is based on a polynomial time algorithm for deciding the existence of $+1$ and -1 cycles in a weighted graph and states five necessary and sufficient conditions for synchronizability. All conditions are verifiable in polynomial time, despite the fact that the length of the resulting synchronization strategy is unbounded (as it depends on the initial weight values). The presented techniques are general and allow for a straightforward adaptation to the situation when more observations become available. Future research will include nontrivial extensions to nondeterministic WA and synchronization under safety constraints, e.g. constraints on the weight-levels encountered during the synchronization.

Acknowledgments. The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement 601148 (CASSTING), EU FP7 FET project SENSATION, Sino-Danish Basic Research Center IDAE4CPS, the European Research Council (ERC) under grant agreement 267989 (QUAREM), the Austrian Science Fund (FWF) project S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award), the Czech Science Foundation under grant agreement P202/12/G061, and People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) REA Grant No 291734.

Paper C

Average-energy Games

Simon Laursen Kim G. Larsen
Aalborg University, Department of Computer Science, Denmark

Patricia Bouyer Nicolas Markey Mickael Randour
LSV – CNRS & ENS Cachan, France

Mickael Randour
ULB - Université libre de Bruxelles, Belgium

Abstract Two-player quantitative zero-sum games provide a natural framework to synthesize controllers with performance guarantees for reactive systems within an uncontrollable environment. Classical settings include mean-payoff games, where the objective is to optimize the long-run average gain per action, and energy games, where the system has to avoid running out of energy. We study *average-energy* games, where the goal is to optimize the long-run average of the accumulated energy. We show that this objective arises naturally in several applications, and that it yields interesting connections with previous concepts in the literature. We prove that deciding the winner in such games is in $\mathsf{NP} \cap \mathsf{coNP}$ and at least as hard as solving mean-payoff games, and we establish that memoryless strategies suffice to win. We also consider the case where the system has to minimize the average-energy *while* maintaining the accumulated energy within predefined bounds at all times: this corresponds to operating with a finite-capacity storage for energy. We give results for one-player and two-player games, and establish complexity bounds and memory requirements.

Publication History The paper was published in the Proceedings the Sixth International Symposium on Games, Automata, Logics and Formal Verification, (GandALF 2015), EPTCS volume 193, pp. 1–15, 2015 by the Open Publishing Association. The full version was published in the journal Acta Informatica July 2016, pp. 1–37. This thesis includes the full version of the paper with all proves and a modified layout.

The layout has been revised.

C.1 Introduction

Quantitative games. Game-theoretic formulations are a standard tool for the synthesis of provably-correct controllers for reactive systems [70]. We consider two-player (system vs. environment) turn-based games played on finite graphs. Vertices of the graph are called *states* and partitioned into states of player 1 and states of player 2. The game is played by moving a pebble from state to state, along *edges* in the graph, and starting from a given initial state. Whenever the pebble is on a state belonging to player i, player i decides where to move the pebble next, according to his *strategy*. The infinite path followed by the pebble is called a *play*: it represents one possible behavior of the system. A *winning objective* encodes acceptable behaviors of the system and can be seen as a set of winning plays. The goal of player 1 is to ensure that the outcome of the game will be a winning play, whatever the strategy played by his adversary.

To reason about resource constraints and the performance of strategies, *quantitative games* have been considered in the literature. See for example [31, 12, 111], or [112] for an overview. Those games are played on *weighted* graphs, where edges are fitted with integer weights modeling rewards or costs. The performance of a play is evaluated via a *payoff function* that maps it to the numerical domain. The objective of player 1 is then to ensure a sufficient payoff with regard to a given threshold value. Seminal classes of quantitative games include mean-payoff (MP), total-payoff (TP) and energy games (EG). In MP games [54, 126, 79], player 1 has to optimize his long-run average gain per edge taken whereas, in TP games [68, 66], player 1 has to optimize his long-run sum of weights. Energy games [31, 18, 78] model safety-like properties: the goal is to ensure that the running sum of weights never drops below zero and/or that it never exceeds a given upper bound $U \in \mathbb{N}$. All three classes share common properties. First, MP games, TP games, and EG games with only a lower bound ($\mathsf{Energy}_{\mathsf{L}}$) are memoryless determined (given an initial state, either player 1 has a strategy to win, or player 2 has one, and in both cases no memory is required to win). Second, deciding the winner for those games is in $\mathsf{NP} \cap \mathsf{coNP}$ and no polynomial algorithm is known despite many efforts (e.g., [26, 33]). Energy games with both lower and upper bounds ($\mathsf{Energy}_{\mathsf{LU}}$) are more complex: they are $\mathsf{EXPTIME}$-complete and winning requires memory in general [18].

While those classes are well-known, it is sometimes necessary to go beyond them to accurately model practical applications. For example, multi-dimensional games and conjunctions with a parity objective model trade-offs between

different quantitative aspects [38, 36, 121]. Similarly, window objectives address the need for strategies ensuring good quantitative behaviors within reasonable time frames [33].

Average-energy games. We study the *average-energy* (AE) payoff function: in AE games, the goal of player 1 is to optimize the *long-run average accumulated energy* over a play. We introduce this objective to formalize the specification desired in a practical application [29], which we detail in the following as a motivating example. Interestingly, it turns out that this payoff first appeared long ago [120], but it was not subject to a systematic study until very recently: see related work for more discussion.

In addition to being meaningful w.r.t. practical applications, AE games also have theoretical interest. In [35], Chatterjee and Prabhu define the *average debit-sum level* objective, which can be seen as a variation of the *average-energy* where the accumulated energy is taken to be zero in any point where it is actually positive (hence, it focuses on the average debt). They use the corresponding games to compute the values of quantitative timed simulation functions. In particular, they provide a pseudo-polynomial-time algorithm to solve those games, but the complexity of deciding the winner as well as the memory requirements are open. Here, we solve those questions for the very similar average-energy objective.

Motivating example. Our example is a simplified version of the industrial application studied by Cassez et al. in [29]. Consider a machine that consumes oil, stored in a connected accumulator. We want to synthesize an appropriate controller to operate the oil pump that fills the accumulator, and by the effect of pressure, that releases oil from the accumulator into the machine with a (time-varying) rate according to desired production. In order to ensure safety, the oil level in the accumulator should be maintained at all times between a minimal and a maximal level. This part of the specification can be encoded as an energy objective with both lower and upper bounds ($\mathsf{Energy}_{\mathsf{LU}}$). At the same time, the more oil (thus pressure) in the accumulator, the faster the whole apparatus wears out. Hence, an ideal controller should minimize the average level of oil in the long run. This desire can be formalized through the average-energy payoff (AE). Overall, the specification is thus to minimize the average-energy under the strong energy constraints: we denote the corresponding objective by $\mathsf{AvgEnergy}_{\mathsf{LU}}$.

Contributions. Our main results are summarized in Table C.1.

Game objective	1-player	2-player	memory
MP	PTIME [82]	NP ∩ coNP [126]	memoryless [54]
TP	PTIME [62]	NP ∩ coNP [66]	memoryless [68]
$\mathsf{Energy}_{\mathsf{L}}$	PTIME [18]	NP ∩ coNP [31, 18]	memoryless [31]
$\mathsf{Energy}_{\mathsf{LU}}$	PSPACE-c. [61]	EXPTIME-c. [18]	pseudo-polynomial
AE	PTIME	NP ∩ coNP	memoryless
$\mathsf{AvgEnergy}_{\mathsf{LU}}$, polynomial U	PTIME	NP ∩ coNP	polynomial
$\mathsf{AvgEnergy}_{\mathsf{LU}}$, arbitrary U	PSPACE-c.	EXPTIME-c.	pseudo-polynomial
$\mathsf{AvgEnergy}_{\mathsf{L}}$	PSPACE-e. / NP-h.	*open* / EXPTIME-h.	*open* ($\geq$ pseudo-p.)

Table C.1: Complexity of deciding the winner and memory requirements for quantitative games: MeanPayoff stands for mean-payoff, TP for total-payoff, $\mathsf{Energy}_{\mathsf{L}}$ (resp. $\mathsf{Energy}_{\mathsf{LU}}$) for lower-bounded (resp. lower- and upper-bounded) energy, AE for average-energy, $\mathsf{AvgEnergy}_{\mathsf{L}}$ (resp. $\mathsf{AvgEnergy}_{\mathsf{LU}}$) for average-energy under a lower bound (resp. and upper bound $U \in \mathbb{N}$) on the energy, c. for complete, e. for easy, and h. for hard. Results without reference are proved in this paper.

A) We establish that the average-energy objective can be seen as a *refinement* of total-payoff, in the same sense as total-payoff is seen as a refinement of mean-payoff [66]: it allows to distinguish strategies yielding identical mean-payoff and total-payoff.

B) We show that deciding the winner in two-player AE games is in NP ∩ coNP whereas it is in PTIME for one-player games. In both cases, memoryless strategies suffice (Thm. C.8). Those complexities match the state-of-the-art for MeanPayoff and TP games [126, 79, 66, 26]. Furthermore we prove that AE games are at least as hard as mean-payoff games (Thm. C.10). Therefore, the NP ∩ coNP-membership can be considered optimal w.r.t. our knowledge of MeanPayoff games. Technically, the crux of our approach is as follows. First, we show that memoryless strategies suffice in one-player AE games (Thm. C.6): this requires to prove important properties of the AE payoff as classical sufficient criteria for memoryless determinacy present in the literature fail to apply directly. Second, we establish a polynomial-time algorithm for the one-player case: it exploits the structure of winning strategies and mixes graph techniques with local linear program solving (Thm. C.7). Finally, we lift memoryless determinacy to the two-player case using results by Gimbert and Zielonka [69] and obtain the NP ∩ coNP-membership as a corollary (Thm. C.9).

C) We establish an EXPTIME algorithm to solve two-player AE games with lower- and upper-bounded energy ($\mathsf{AvgEnergy}_{\mathsf{LU}}$) with an arbitrary upper bound $U \in \mathbb{N}$ (Thm. C.13). It relies on a reduction of the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game to a pseudo-polynomially larger AE game where the energy constraints are

encoded in the graph structure. Applying straightforwardly the AE algorithm on this game would only give us NEXPTIME ∩ coNEXPTIME-membership, hence we avoid this blowup by further reducing the problem to a particular MeanPayoff game and applying a pseudo-polynomial algorithm, with some care to ensure that overall the algorithm only requires pseudo-polynomial time in the original $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game. Since the simpler $\mathsf{Energy}_{\mathsf{LU}}$ games (i.e., $\mathsf{AvgEnergy}_{\mathsf{LU}}$ with a trivial AE constraint) are already EXPTIME-hard [18], the EXPTIME-membership result is optimal. We also prove that pseudo-polynomial memory is both sufficient and in general necessary to win in $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games, for both players (Thm. C.15). We show that one-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games are PSPACE-complete via the on-the-fly construction of a witness path based on the aforementioned reduction, answering a question left open in [19]. For polynomial (in the size of the game graph) values of the upper bound U — or if it is given in unary — the complexity of the two-player (resp. one-player) $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem collapses to NP ∩ coNP (resp. PTIME) with the same approach, and polynomial memory suffices for both players.

D) We provide partial answers for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective — AE under a lower bound constraint on energy but no upper bound. We show PSPACE-membership for the one-player case (Thm. C.18), by reducing the problem to an $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game with a sufficiently large upper bound. That is, we prove that if the player can win for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective, then he can do so without ever increasing its energy above a well-chosen bound. We also prove the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem to be at least NP-hard in one-player games (Thm. C.18) and EXPTIME-hard in two-player games (Lem. C.21) via reductions from the subset-sum problem and countdown games respectively. Finally, we show that memory is required for both players in two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games (Lem. C.22), and that pseudo-polynomial memory is both sufficient and necessary in the one-player case (Thm. C.19). The decidability status of two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games remains open as we only provide a correct but incomplete incremental algorithm (Lem. C.20). We conjecture that the two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem is decidable and sketch a potential approach to solve it. We highlight the key remaining questions and discuss some connections with related models that are known to be difficult.

Observe that in many applications, the energy must be stocked in a finite-capacity storage for which an upper bound is provided. Hence, the model of choice in this case is $\mathsf{AvgEnergy}_{\mathsf{LU}}$.

Related work. This paper extends previous work presented in a conference [19]: it gives a full presentation of the technical details, along with

additional results and improved complexities.

The *average-energy* payoff — Eq. (C.1) — appeared in a paper by Thuijsman and Vrieze in the late eighties [120], under the name *total-reward*. This definition is different from the classical *total-payoff* — see Sect. C.2 — commonly studied in the formal methods community (see for example [68, 66]), which, despite that, has been referred in many papers as either total-payoff or total-reward equivalently. We will see in this paper that both definitions are *indeed* different and exhibit different behaviors.

Maybe due to this confusion, the payoff of Eq. (C.1) — which we call *average-energy* thus avoiding misunderstandings — was not studied extensively until recently. Nothing was known about memoryless determinacy and complexity of deciding the winner. Independently to our work, Boros et al. recently studied the same payoff (under the name *total-payoff*). In [17], they study Markov decision processes and stochastic games with the payoff of Eq. (C.1) and solve both questions. Their results overlap with ours for AE games (Table C.1). Let us first mention that our results were obtained independently. Second, and *most importantly*, our approach and *techniques are different*, and we believe our take on the problem yields some interest for our community. Indeed, the algorithm of Boros et al. entirely relies on linear programming in the one-player case, and resorts to approximation by discounted games in the two-player one. Our techniques are arguably more constructive and based on inherent properties of the payoff. In that sense, it is closer to what is usually deemed important in our field. For example, we provide an extensive comparison with classical payoffs. We base our proof of memoryless determinacy on *operational understanding* of the AE which is crucial in order to formalize proper specifications. Our technique then benefits from seminal works [69] to bypass the reduction to discounted games and obtain a direct proof, thanks to our more constructive approach. Lastly, while [17] considers the AE problem in the stochastic context, we focus on the deterministic one but consider multi-criteria extensions by adding bounds on the energy ($\mathsf{AvgEnergy}_{\mathsf{LU}}$ and $\mathsf{AvgEnergy}_{\mathsf{L}}$ games). Those extensions are *completely new*, exhibit theoretical interest and are adequate for practical applications in constrained energy systems, as witnessed by the case study of [29].

Recent work of Brázdil et al. [23] considers the optimization of a payoff under energy constraint. They study mean-payoff in consumption systems, i.e., simplified one-player energy games where all edges consume energy but some states can atomically produce a reload of the energy up to the allowed capacity.

C.2 Preliminaries

Graph games. We consider turn-based games played on graphs between two players denoted by $\mathcal{P}_1$ and $\mathcal{P}_2$. A *game* is a tuple $G(T) = (S_1, S_2, E, w)$ where (i) S_1 and S_2 are disjoint finite sets of *states* belonging to $\mathcal{P}_1$ and $\mathcal{P}_2$, with $S = S_1 \uplus S_2$, (ii) $E \subseteq S \times S$ is a finite set of *edges*, and (iii) $w\colon E \to \mathbb{Z}$ is an integer *weight function*. Given edge $(s_1, s_2) \in E$, we write $w(s_1, s_2)$ as a shortcut for $w((s_1, s_2))$. We denote by W the largest absolute weight assigned by function w. A game is called 1-player if $S_1 = \emptyset$ or $S_2 = \emptyset$.

A *play* from an initial state $s_{\mathsf{init}} \in S$ is an infinite sequence $\pi = s_0 s_1 \dots s_n \dots$ such that $s_0 = s_{\mathsf{init}}$ and for all $i \geq 0$ we have $(s_i, s_{i+1}) \in E$. The (finite) *prefix* of π up to position n gives the sequence $\pi(n) = s_0 s_1 \dots s_n$, the first (resp. last) element s_0 (resp. s_n) is denoted $\mathit{first}(\pi(n))$ (resp. $\mathit{last}(\pi(n))$). The set of all plays in $G(T)$ is denoted by $\mathit{Plays}(G(T))$ and the set of all prefixes is denoted by $\mathit{Prefs}(G(T))$. We say that a prefix $\rho \in \mathit{Prefs}(G(T))$ belongs to $\mathcal{P}_i$, $i \in \{1, 2\}$, if $\mathit{last}(\rho) \in S_i$. The set of prefixes that belong to $\mathcal{P}_i$ is denoted by $\mathit{Prefs}_i(G(T))$. The classical concatenation between prefixes (resp. prefix and play) is denoted by the $\cdot$ operator. The length of a non-empty prefix $\rho = s_0 \dots s_n$ is defined as the number of edges and denoted by $|\rho| = n$.

Payoffs of plays. Given a play $\pi = s_0 s_1 \dots s_n \dots$ we define

- its *energy level* at position n as

$$\mathrm{EL}(\pi(n)) = \sum_{i=0}^{n-1} w(s_i, s_{i+1});$$

- its *mean-payoff* as

$$\overline{MP}(\pi) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} w(s_i, s_{i+1}) = \limsup_{n\to\infty} \frac{1}{n} \mathrm{EL}(\pi(n));$$

- its *total-payoff* as

$$\overline{TP}(\pi) = \limsup_{n\to\infty} \sum_{i=0}^{n-1} w(s_i, s_{i+1}) = \limsup_{n\to\infty} \mathrm{EL}(\pi(n));$$

- and its *average-energy* as

$$\overline{AE}(\pi) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=0}^{i-1} w(s_j, s_{j+1}) \right) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathrm{EL}(\pi(i)). \tag{C.1}$$

We will sometimes consider those measures defined with lim inf instead of lim sup, in which case we write $\underline{MP}$, $\underline{TP}$ and $\underline{AE}$ respectively. Finally, we also consider those measures over prefixes: we naturally define them by dropping the $\limsup_{n\to\infty}$ operator and taking $n = |\rho|$ for a prefix $\rho \in \mathit{Prefs}(G(T))$. In this case, we simply write $\mathsf{MeanPayoff}(\rho)$, $TP(\rho)$ and $AE(\rho)$ to denote the fact that we consider *finite* sequences.

Strategies. A *strategy* for $\mathcal{P}_i$, $i \in \{1, 2\}$, is a function $\sigma_i \colon \mathit{Prefs}_i(G(T)) \to S$ such that for all $\rho \in \mathit{Prefs}_i(G(T))$ we have $(\mathit{last}(\rho), \sigma_i(\rho)) \in E$. A strategy σ_i for $\mathcal{P}_i$ is *finite-memory* if it can be encoded by a deterministic Moore machine $(M, m_0, \alpha_u, \alpha_n)$ where M is a finite set of states (the memory of the strategy), $m_0 \in M$ is the initial memory state, $\alpha_u \colon M \times S \to M$ is an update function, and $\alpha_n \colon M \times S_i \to S$ is the next-action function. If the game is in $s \in S_i$ and $m \in M$ is the current memory value, then the strategy chooses $s' = \alpha_n(m, s)$ as the next state of the game. When the game leaves a state $s \in S$, the memory is updated to $\alpha_u(m, s)$. Formally, $(M, m_0, \alpha_u, \alpha_n)$ defines the strategy σ_i such that $\sigma_i(\rho \cdot s) = \alpha_n(\hat{\alpha}_u(m_0, \rho), s)$ for all $\rho \in S^*$ and $s \in S_i$, where $\hat{\alpha}_u$ extends α_u to sequences of states as expected. A strategy is *memoryless* if $|M| = 1$, i.e., it does not depend on the history but only on the current state of the game. We denote by $\Sigma_i(G(T))$, the sets of strategies for player $\mathcal{P}_i$. We drop $G(T)$ when the context is clear.

A play $\pi = s_0 s_1 \dots$ is *consistent* with a strategy σ_i of $\mathcal{P}_i$ if, for all $n \geq 0$ where $\mathit{last}(\pi(n)) \in S_i$, we have $\sigma_i(\pi(n)) = s_{n+1}$. Given an initial state $s_{\mathsf{init}} \in S$ and strategies σ_1 and σ_2 for the two players, we denote by $\mathit{Outcome}(s_{\mathsf{init}}, \sigma_1, \sigma_2)$ the unique play that starts in s_{init} and is consistent with both σ_1 and σ_2. When fixing the strategy of only $\mathcal{P}_i$, we denote the set of consistent outcomes by $\mathit{Outcomes}(s_{\mathsf{init}}, \sigma_i)$.

Objectives. An *objective* in $G(T)$ is a set $\mathcal{W} \subseteq \mathit{Plays}(G(T))$ that is declared winning for $\mathcal{P}_1$. Given a game $G(T)$, an initial state s_{init}, and an objective $\mathcal{W}$, a strategy $\sigma_1 \in \Sigma_1$ is winning for $\mathcal{P}_1$ if for all strategy $\sigma_2 \in \Sigma_2$, we have that $\mathit{Outcome}(s_{\mathsf{init}}, \sigma_1, \sigma_2) \in \mathcal{W}$. Symmetrically, a strategy $\sigma_2 \in \Sigma_2$ is winning for $\mathcal{P}_2$ if for all strategy $\sigma_1 \in \Sigma_1$, we have that $\mathit{Outcome}(s_{\mathsf{init}}, \sigma_1, \sigma_2) \notin \mathcal{W}$. That is, we consider *zero-sum* games.

We consider the following objectives and combinations of those objectives.

- Given an initial energy level $c_{\mathsf{init}} \in \mathbb{N}$, the **lower-bounded energy** ($\mathsf{Energy_L}$) objective $\mathit{Energy}_L(c_{\mathsf{init}}) = \{\pi \in \mathit{Plays}(G) \mid \forall\, n \geq 0,\ c_{\mathsf{init}} + \mathrm{EL}(\pi(n)) \geq 0\}$ requires non-negative energy at all times.

- Given an upper bound $U \in \mathbb{N}$ and an initial energy level $c_{\mathsf{init}} \in \mathbb{N}$, the **lower- and upper-bounded energy** ($\mathsf{Energy}_{\mathsf{LU}}$) objective $\mathit{Energy}_{LU}(U, c_{\mathsf{init}}) = \{\pi \in \mathit{Plays}(G(T)) \mid \forall\, n \geq 0,\ c_{\mathsf{init}} + \mathrm{EL}(\pi(n)) \in [0, U]\}$ requires that the energy always remains non-negative and below the upper bound U along a play.

- Given a threshold $t \in \mathbb{Q}$, the **mean-payoff** (MP) objective $\mathit{MeanPayoff}(t) = \{\pi \in \mathit{Plays}(G(T)) \mid \overline{MP}(\pi) \leq t\}$ requires that the mean-payoff is at most t.

- Given a threshold $t \in \mathbb{Z}$, the **total-payoff** (TP) objective $\mathit{TotalPayoff}(t) = \{\pi \in \mathit{Plays}(G(T)) \mid \overline{TP}(\pi) \leq t\}$ requires that the total-payoff is at most t.

- Given a threshold $t \in \mathbb{Q}$, the **average-energy** (AE) objective $\mathit{AvgEnergy}(t) = \{\pi \in \mathit{Plays}(G(T)) \mid \overline{AE}(\pi) \leq t\}$ requires that the average-energy is at most t.

For the MP, TP and AE objectives, note that $\mathcal{P}_1$ aims to *minimize* the payoff value while $\mathcal{P}_2$ tries to maximize it. The reversed convention is also often used in the literature but both are equivalent. For our motivating example, seeing $\mathcal{P}_1$ as a minimizer is more natural. Note that we define the objectives using the lim sup variants of MP, TP and AE, but similar results are obtained for the lim inf variants.

Decision problem. Given a game $G(T)$, an initial state $s_{\mathsf{init}} \in S$, and an objective $\mathcal{W} \subseteq \mathit{Plays}(G(T))$ as defined above, the associated *decision problem* is to decide if $\mathcal{P}_1$ has a winning strategy for this objective.

We recall classical results in Table C.1. Memoryless strategies suffice for both players for $\mathsf{Energy}_{\mathsf{L}}$ [31, 18], MP [54] and TP [62, 68] objectives. Since all associated problems can be solved in polynomial time for 1-player games, it follows that the 2-player decision problem is in $\mathsf{NP} \cap \mathsf{coNP}$ for those three objectives [18, 126, 66]. For the $\mathsf{Energy}_{\mathsf{LU}}$ objective, memory is in general needed and the associated decision problem is $\mathsf{EXPTIME}$-complete [18] (PSPACE-complete for one-player games [61]).

Game values. Given a game with an objective $\mathcal{W} \in \{\mathit{MeanPayoff}, \mathit{TotalPayoff}, \mathit{AvgEnergy}\}$ and an initial state s_{init}, we refer to the *value* from s_{init} as $v = \inf\{t \in \mathbb{Q} \mid \exists\, \sigma_1 \in \Sigma_1,\ \mathit{Outcomes}(s_{\mathsf{init}}, \sigma_1) \subseteq \mathcal{W}(t)\}$. For both MP and TP objectives, it is known that the value can be achieved

by an optimal memoryless strategy; for the AE objective it follows from our results (Thm. C.8).

C.3 Average-Energy

In this section, we consider the problem of ensuring a *sufficiently low* average-energy.

Problem C.1 (AE)**.** *Given a game $G(T)$, an initial state s_{init}, and a threshold $t \in \mathbb{Q}$, decide if $\mathcal{P}_1$ has a winning strategy $\sigma_1 \in \Sigma_1$ for the objective $AvgEnergy(t)$.*

We first compare the AE objective with traditional quantitative objectives and study how they can be connected (Sect. C.3.1). Then we want to establish that in AE games, memoryless strategies are always sufficient to play optimally, for *both* players. Interestingly, this result cannot be obtained by straightforward application of many well-known sufficient criteria for memoryless determinacy existing in the literature. We thus introduce some technical lemmas that highlight the inherent features of the AE payoff function (Sect. C.3.2) and permit to prove the result for *one-player* AE games (Sect. C.3.3). We then prove that one-player AE games can be solved in polynomial-time via an algorithm combining graph analysis techniques with linear programming. Finally, we consider the *two-player* case (Sect. C.3.4). Applying a result by Gimbert and Zielonka [69], combined with our results on the one-player case, we derive memoryless determinacy of two-player AE games. This also induces $\mathsf{NP} \cap \mathsf{coNP}$-membership of the AE problem by the PTIME algorithm of Sect. C.3.3. We conclude by proving that AE games are at least as hard as MP games, hence indicating that the $\mathsf{NP} \cap \mathsf{coNP}$ upper bound is essentially optimal with regard to our current knowledge of MP games (whose membership to PTIME is a long-standing open problem [126, 79, 26, 33]).

C.3.1 Relation with Classical Objectives

Several links between $\mathsf{Energy}_{\mathsf{L}}$, MP and TP objectives can be established. Intuitively, $\mathcal{P}_1$ can only ensure a lower bound on energy if he can prevent $\mathcal{P}_2$ from enforcing strictly-negative cycles (otherwise the initial energy is eventually exhausted). This is the case if and only if $\mathcal{P}_1$ can ensure a non-negative mean-payoff in $G(T)$ (here, he wants to maximize the $\mathsf{MeanPayoff}$), and if this is the case, $\mathcal{P}_1$ can prevent the running sum of weights from ever going too far

beyond zero along a play, hence granting a lower bound on total-payoff. We introduce the sign-reversed game $G(T)'$ in the next lemma by consistency with our view of $\mathcal{P}_1$ as a minimizer with regard to payoffs (as discussed in Sect. C.2).

Lemma C.1. *Let $G(T) = (S_1, S_2, E, w)$ be a game and $s_{\mathsf{init}} \in S$ be the initial state. The following assertions are equivalent.*

A. *There exists $c_{\mathsf{init}} \in \mathbb{N}$ such that $\mathcal{P}_1$ has a (memoryless) winning strategy for objective $Energy_L(c_{\mathsf{init}})$.*

B. *Player $\mathcal{P}_1$ has a (memoryless) winning strategy for objective $MeanPayoff(0)$ in the game $G(T)'$ defined by reversing the sign of the weight function, i.e., for all $(s_1, s_2) \in E$, $w'(s_1, s_2) = -w(s_1, s_2)$.*

C. *Player $\mathcal{P}_1$ has a (memoryless) winning strategy for objective $TotalPayoff(t)$, with $t = 2 \cdot (|S| - 1) \cdot W$, in the game $G(T)'$ defined by reversing the sign of the weight function.*

D. *There exists $t \in \mathbb{Z}$ such that $\mathcal{P}_1$ has a (memoryless) winning strategy for objective $TotalPayoff(t)$, in the game $G(T)'$ defined by reversing the sign of the weight function.*

Proof. Proof of $A \Leftrightarrow B$ is given in [18, Proposition 12]. Proof of $B \Leftrightarrow C \Leftrightarrow D$ is in [33, Lem. 1]. □

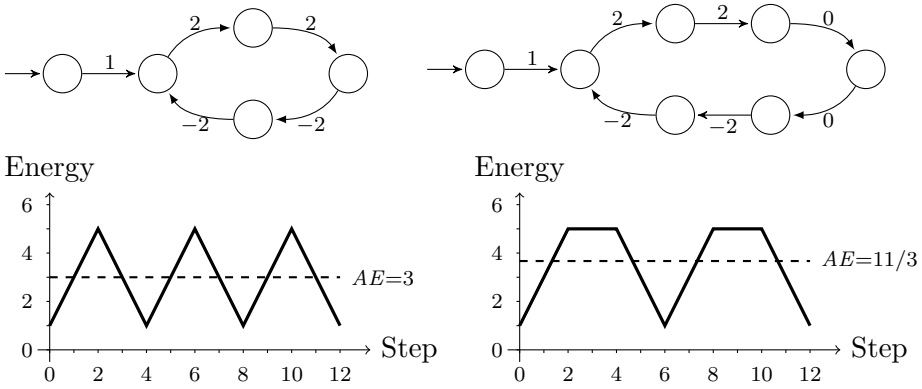
The *TP* objective is sometimes seen as a *refinement* of *MP* for the case where $\mathcal{P}_1$ — as a minimizer — can ensure MeanPayoff equal to zero but not lower, i.e., the *MP* game has value zero [66]. Indeed, one may use the *TP* to further discriminate between strategies that guarantee MeanPayoff zero. In the same philosophy, the average-energy can help in distinguishing strategies that yield identical total-payoffs. See Fig. C.1. The *AE* values in both examples can be computed easily using the upcoming technical lemmas (Sect. C.3.2).

In these examples, the average-energy is clearly comprised between the infimum and supremum total-payoffs. This remains true for any play.

Lemma C.2. *For any play $\pi \in Plays(G(T))$, we have that*

$$\underline{AE}(\pi), \overline{AE}(\pi) \in \left[\underline{TP}(\pi), \overline{TP}(\pi)\right] \subseteq \mathbb{R} \cup \{-\infty, \infty\}.$$

Proof. Consider a play $\pi \in Plays(G(T))$. By definition of the total-payoff and thanks to weights taking integer values, we have that there exists some index $m \in \mathbb{N}_0$ such that, for all $n \geq m$, $\mathrm{EL}(\pi(n)) \in \left[\underline{TP}(\pi), \overline{TP}(\pi)\right]$. By definition, the average-energy $\overline{AE}$ (resp. $\underline{AE}$) measures the supremum (resp. infinimum)



(c) Play π_1 sees energy levels $(1,3,5,3)^\omega$.

(d) Play π_2 sees energy levels $(1,3,5,5,5,3)^\omega$.

Figure C.1: Both plays have identical mean-payoff and total-payoff: $\overline{MP}(\pi_1) = \underline{MP}(\pi_1) = \overline{MP}(\pi_2) = \underline{MP}(\pi_2) = 0$, $\overline{TP}(\pi_1) = \overline{TP}(\pi_2) = 5$, and $\underline{TP}(\pi_1) = \underline{TP}(\pi_2) = 1$. But play π_1 has a lower average-energy: $\overline{AE}(\pi_1) = \underline{AE}(\pi_1) = 3 < \overline{AE}(\pi_2) = \underline{AE}(\pi_2) = 11/3$.

limit of the averages of those partial sums, hence it holds that $\underline{AE}(\pi), \overline{AE}(\pi) \in \left[\underline{TP}(\pi), \overline{TP}(\pi)\right]$. □

In particular, *if the mean-payoff value from a state is not zero*, its total-payoff value is infinite and the following lemma holds.

Lemma C.3. *Let $G(T) = (S_1, S_2, E, w)$ be a game and $s_{\mathsf{init}} \in S$ be the initial state.*

1. *If there exists $t < 0$ such that $\mathcal{P}_1$ has a (memoryless) winning strategy for objective MeanPayoff(t), then $\mathcal{P}_1$ has a memoryless strategy that is winning for AvgEnergy(t') for all $t' \in \mathbb{Q}$, i.e., this strategy ensures that any consistent outcome π is such that $\underline{AE}(\pi) = \overline{AE}(\pi) = -\infty$.*
2. *If $\mathcal{P}_1$ has no (memoryless) winning strategy for MeanPayoff(0), then, for any $t' \in \mathbb{Q}$, $\mathcal{P}_1$ has no winning strategy for AvgEnergy(t'). In particular, $\mathcal{P}_2$ has a memoryless strategy ensuring that any consistent outcome π is such that $\underline{AE}(\pi) = \overline{AE}(\pi) = \infty$.*

Proof. Consider the first implication. Assume $\mathcal{P}_1$ has a memoryless strategy σ_1 ensuring that all outcomes $\pi \in \mathit{Outcomes}(s_{\mathsf{init}}, \sigma_1)$ are such that $\overline{MP}(\pi) < 0$. For any such outcome, it is guaranteed that all simple cycles have a strictly negative energy level. Thus, we have that $\overline{TP}(\pi) = -\infty$, and by Lem. C.2, it implies that $\overline{AE}(\pi) = -\infty$, as claimed. Since $\underline{AE}(\pi) \leq \overline{AE}(\pi)$ by definition,

the property holds.

Now consider the second implication. Assume there exists no winning strategy for $\mathcal{P}_1$ for the mean-payoff objective. By equivalence B $\Leftrightarrow$ D of Lem. C.1, and memoryless determinacy of total-payoff games (see for example [68]), it follows that $\mathcal{P}_2$ has a memoryless strategy σ_2 ensuring that all consistent outcomes $\pi \in \mathit{Outcomes}(s_{\mathsf{init}}, \sigma_2)$ are such that $\underline{TP}(\pi) = \infty$. By Lem. C.2, this induces the claim. □

C.3.2 Useful Properties of the Average-energy

In this subsection, we will first review some classical criteria that usually prove sufficient to deduce memoryless determinacy in quantitative games and discuss why they cannot be applied straight out of the box to the average-energy payoff. We will then prove two useful properties of this payoff that will later help us to prove the desired result.

Classical sufficient criteria. We briefly discuss traditional approaches to prove memoryless determinacy in quantitative games. The first one is to study a variant of the infinite-duration game where the game halts as soon as a cycle is closed and then to relate the properties of this variant to the infinite-duration game. This technique was used in the original proof of memoryless determinacy for mean-payoff games by Ehrenfeucht and Mycielski [54], and in a following simpler proof by Björklund et al. [11]. The connection between infinite-duration games and so-called *first cycle games* was recently streamlined by Aminof and Rubin [2], identifying sufficient conditions to prove that first cycle games and their infinite-duration counterparts admit optimal memoryless strategies for both players. Among those conditions is the need for winning objectives to be closed under *cyclic permutation* and under *concatenation*. Without further assumptions, the average-energy objective satisfies neither. Indeed, consider cycles represented by sequences of *weights* $\mathcal{C}_1 = \{-1\}$, $\mathcal{C}_2 = \{1\}$ and $\mathcal{C}_3 = \{1, -2\}$. We see that $AE(\mathcal{C}_1\mathcal{C}_2) = (-1+0)/2 = -1/2 < AE(\mathcal{C}_2\mathcal{C}_1) = (1-0)/2 = 1/2$, hence AE is not closed under cyclic permutations. Intuitively, the order in which the weights are seen *does* matter, in contrast to most classical payoffs. For concatenation, see that $AE(\mathcal{C}_3) = 0$ while $AE(\mathcal{C}_3\mathcal{C}_3) = -1/2 < 0$. Here the intuition is that the overall AE is impacted by the energy of the first cycle which is strictly negative (-1). In a sense, the AE of a cycle can only be maintained through repetition if this cycle is neutral with regard to the total energy level, i.e., if it has energy level zero: we will formalize this intuition in Lem. C.5.

Other criteria for memoryless determinacy or half-memoryless determinacy (i.e., holding only for one of the two players) respectively appear in works by Gimbert and Zielonka [68] and by Kopczynski [86]. They involve checking that the payoff is *fairly mixing*, or *concave*. Again, both are false for arbitrary sequences of weights in the case of the average-energy, for essentially the same reasons as above. Nevertheless, we will be able to prove that memoryless strategies suffice for both players using similar ideas but first taking care of the problematic cases. Intuitively, when those cases are dealt with, we will regain a payoff that satisfies the above conditions. We also obtain *monotonicity* and *selectivity* of the payoff function as defined in [69].

Extraction of prefixes. The following lemma describes the impact of adding a finite prefix to an infinite play. We prove that the average-energy over a play can be decomposed w.r.t. to the energy level of any of its prefixes and the average-energy of the remaining suffix.

Lemma C.4 (**Average-energy prefix**). *Let* $\rho \in \mathit{Prefs}(G(T))$, $\pi \in \mathit{Plays}(G(T))$. *Then,*

$$\overline{AE}(\rho \cdot \pi) = \mathrm{EL}(\rho) + \overline{AE}(\pi).$$

The same equality holds for $\underline{AE}$.

Proof. Let $\rho = s_0 \dots s_k \in \mathit{Prefs}(G(T))$ and $\pi \in \mathit{Plays}(G(T))$ be a prefix and a play over a game $G(T)$. We prove the property for $\overline{AE}$. By definition and decomposition, we have that

$$\begin{aligned}\overline{AE}(\rho \cdot \pi) &= \limsup_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \mathrm{EL}((\rho \cdot \pi)(i)) \\ &= \limsup_{n\to\infty} \left[\frac{1}{n} \cdot \sum_{i=1}^{k} \mathrm{EL}(\rho(i)) + \frac{1}{n} \cdot \sum_{i=k+1}^{n} \mathrm{EL}(\rho) + \frac{1}{n} \cdot \sum_{i=k+1}^{n} \mathrm{EL}(\pi(i-k)) \right].\end{aligned}$$

For clarity, we rewrite this expression as $\overline{AE}(\rho \cdot \pi) = \limsup_{n\to\infty} \big[X_1(n) + X_2(n) + X_3(n)\big]$, maintaining the same order.

Since k is fixed and finite, and $\mathrm{EL}(\rho(i))$ is bounded for all $i \leq k$, we have that $\limsup_{n\to\infty} X_1(n) = \lim_{n\to\infty} X_1(n) = 0$. Furthermore, for $n \geq k+1$, we rewrite the second term as $X_2(n) = (n-k-1) \cdot \mathrm{EL}(\rho)/n$, and it follows that $\limsup_{n\to\infty} X_2(n) = \lim_{n\to\infty} X_2(n) = \mathrm{EL}(\rho)$. Since both sequences $X_1(n)$ and

$X_2(n)$ converge, we can write

$$\liminf_{n\to\infty} X_1(n) + \liminf_{n\to\infty} X_2(n) + \limsup_{n\to\infty} X_3(n) \leq \overline{AE}(\rho\cdot\pi) \leq \limsup_{n\to\infty} X_1(n) + \limsup_{n\to\infty} X_2(n) + \limsup_{n\to\infty} X_3(n).$$

Hence, by a small change of variable,

$$\overline{AE}(\rho\cdot\pi) = \mathrm{EL}(\rho) + \limsup_{n\to\infty} X_3(n) = \mathrm{EL}(\rho) + \limsup_{n\to\infty}\left[\frac{1}{n}\cdot\sum_{i=1}^{n-k-1}\mathrm{EL}(\pi(i))\right] = \mathrm{EL}(\rho) + \overline{AE}(\pi),$$

as, in the limit, the $(k+1)$ missing terms in the sum are negligible. The proof for $\underline{AE}$ is similar. □

Extraction of a best cycle. The next lemma is crucial to prove that memoryless strategies suffice: under well-chosen conditions, one can always select a best cycle in a play — hence, there is no interest in mixing different cycles and no use for memory. It holds only for sequences of cycles *that have energy level zero*: since they do not change the energy, they do not modify the AE of the following suffix of play, and one can decompose the AE as a weighted average over zero cycles.

Lemma C.5 (Repeated zero cycles of bounded length). *Let $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \ldots$ be an infinite sequence of cycles $\mathcal{C}_i \in Prefs(G(T))$ such that (i) $\pi = \mathcal{C}_1\cdot\mathcal{C}_2\cdot\mathcal{C}_3\cdots \in Plays(G(T))$,[1] (ii) $\forall\, i \geq 1$, $\mathrm{EL}(\mathcal{C}_i) = 0$ and (iii) $\exists\, \ell \in \mathbb{N}*$ such that $\forall\, i \geq 1$, $|\mathcal{C}_i| \leq \ell$. Then the following properties hold.*

1. *The average-energy of π is the weighted average of the average-energies of the cycles:*

$$\overline{AE}(\pi) = \limsup_{k\to\infty}\left[\frac{\sum_{i=1}^{k}|\mathcal{C}_i|\cdot AE(\mathcal{C}_i)}{\sum_{i=1}^{k}|\mathcal{C}_i|}\right]. \tag{C.2}$$

2. *For any cycle $\mathcal{C} \in Prefs(G(T))$ such that $\mathrm{EL}(\mathcal{C}) = 0$, we have that $\overline{AE}(\mathcal{C}^\omega) = AE(\mathcal{C})$.*

3. *Repeating the best cycle gives the lowest AE:*

$$\inf_{i\in\mathbb{N}*} AE(\mathcal{C}_i) = \inf_{i\in\mathbb{N}*}\overline{AE}((\mathcal{C}_i)^\omega) \leq \overline{AE}(\pi)$$

.

[1] We slightly abuse the notation as we see cycles as sequences of *edges*. The concatenation of cycles $\mathcal{C}_a = s\,s'\ldots s$ and $\mathcal{C}_b = s\,s''\ldots s$ is to be understood as its natural interpretation $\mathcal{C}_a\cdot\mathcal{C}_b = s\,s'\ldots s\,s''\ldots s$: the origin state s only appears *once* in the middle and not twice as it would with $\mathcal{C}_a$ and $\mathcal{C}_b$ seen as true sequences of states.

Similar properties hold for $\underline{AE}$.

Observe that since we assume a bound $\ell \in \mathbb{N}*$ on the length of cycles, and the game is played on a finite graph, Point 3 of Lem. C.5 does actually allow to select a *best* cycle: the set of possible cycles of length at most ℓ is finite and the infimum is reached, hence can be replaced by a miminum.

Proof. We prove the three points for $\overline{AE}$, similar arguments can be applied for $\underline{AE}$. Consider Point 1. Let $\pi = s_0^1 \dots s_{|\mathcal{C}_1|}^1 s_1^2 \dots s_{|\mathcal{C}_2|}^2 s_1^3 \dots$ where s_j^i denotes the j-th state of cycle $\mathcal{C}_i$, with $\mathcal{C}_1 = s_0^1 \dots s_{|\mathcal{C}_1|}^1$ and for all $i > 1$, $\mathcal{C}_i = s_{|\mathcal{C}_{i-1}|}^{i-1} s_1^i \dots s_{|\mathcal{C}_i|}^i$. Essentially, $s_{|\mathcal{C}_{i-1}|}^{i-1}$ is both the last state of $\mathcal{C}_{i-1}$ and the first one of $\mathcal{C}_i$: it can also be seen as s_0^i and we later use both notations depending on the role we consider for this state. Given index $k \in \mathbb{N}$ of a state s_k in the classical formulation $\pi = s_0 s_1 s_2 \dots$ such that s_k denotes state s_j^i in our new formulation $\pi = s_0^1 \dots s_{|\mathcal{C}_1|}^1 s_1^2 \dots s_{|\mathcal{C}_2|}^2 s_1^3 \dots$, we define $c(k) = i$ and $p(k) = j$, respectively denoting the index of the corresponding cycle and the position of state s_k within this cycle. We can rewrite the definition of the average-energy of π as

$$\begin{aligned}\overline{AE}(\pi) &= \limsup_{n\to\infty} \left[\frac{1}{n} \sum_{k=1}^{n} \mathrm{EL}(\pi(k))\right] \\ &= \limsup_{n\to\infty} \left[\frac{1}{n} \left(\sum_{i=1}^{c(n)-1} \sum_{j=1}^{|\mathcal{C}_i|} \mathrm{EL}(s_0^1 \dots s_j^i) \;+\; \sum_{j=1}^{p(n)} \mathrm{EL}(s_0^1 \dots s_j^{c(n)}) \right)\right]. \quad \text{(C.3)}\end{aligned}$$

Observe that since all cycles are such that $\mathrm{EL}(\mathcal{C}_i) = 0$, we have that $\mathrm{EL}(s_0^1 \dots s_j^i) = \mathrm{EL}(s_0^i \dots s_j^i)$ for all indices $i \in \mathbb{N}*$, $j \in \{1, \dots, |\mathcal{C}_i|\}$. In other words, the energy level in a given position only depends on the current cycle. Hence, for all $i \in \mathbb{N}*$,

$$\sum_{j=1}^{|\mathcal{C}_i|} \mathrm{EL}(s_0^1 \dots s_j^i) = \sum_{j=1}^{|\mathcal{C}_i|} \mathrm{EL}(s_0^i \dots s_j^i) = |\mathcal{C}_i| \cdot AE(\mathcal{C}_i)$$

where the second equality follows by definition of $AE(\mathcal{C}_i)$. Therefore, Eq. (C.3) becomes

$$\overline{AE}(\pi) = \limsup_{n\to\infty} \left[\frac{1}{n} \left(\sum_{i=1}^{c(n)-1} |\mathcal{C}_i| \cdot AE(\mathcal{C}_i) \;+\; \sum_{j=1}^{p(n)} \mathrm{EL}(s_0^{c(n)} \dots s_j^{c(n)}) \right)\right].$$

Recall that, by hypothesis, there exists $\ell \in \mathbb{N}*$ such that for all $i \geq 1$, $|\mathcal{C}_i| \leq \ell$. Observe that the boundedness of cycles length implies that

(a) $p(n) \leq \ell$,

(b) $\sum_{j=1}^{p(n)} \mathrm{EL}(s_0^{c(n)} \dots s_j^{c(n)})$ is bounded,

(c) $\sum_{i=1}^{c(n)-1} |\mathcal{C}_i| \leq n = \sum_{i=1}^{c(n)-1} |\mathcal{C}_i| + p(n) \leq \sum_{i=1}^{c(n)-1} |\mathcal{C}_i| + \ell$.

Combining those three arguments, we obtain that

$$\limsup_{n\to\infty} \left[\frac{\sum_{i=1}^{c(n)-1} |\mathcal{C}_i| \cdot AE(\mathcal{C}_i)}{\sum_{i=1}^{c(n)-1} |\mathcal{C}_i| + \ell} \right] \leq \overline{AE}(\pi) \leq \limsup_{n\to\infty} \left[\frac{\sum_{i=1}^{c(n)-1} |\mathcal{C}_i| \cdot AE(\mathcal{C}_i)}{\sum_{i=1}^{c(n)-1} |\mathcal{C}_i|} \right]$$

Hence,

$$\overline{AE}(\pi) = \limsup_{k\to\infty} \left[\frac{\sum_{i=1}^{k} |\mathcal{C}_i| \cdot AE(\mathcal{C}_i)}{\sum_{i=1}^{k} |\mathcal{C}_i|} \right]$$

as claimed by Point 1.

Now consider Point 2. For any cycle $\mathcal{C} \in \mathit{Prefs}(G(T))$ such that $\mathrm{EL}(\mathcal{C}) = 0$, all three hypotheses *(i)*, *(ii)*, and *(iii)* are clearly satisfied, with $\ell = |\mathcal{C}|$. Hence by Point 1, we have that

$$\overline{AE}(\mathcal{C}^\omega) = \limsup_{k\to\infty} \left[\frac{k \cdot |\mathcal{C}| \cdot AE(\mathcal{C})}{k \cdot |\mathcal{C}|} \right] = AE(\mathcal{C}).$$

Finally, we prove Point 3. The equality straightforwardly follows from Point 2. It remains to consider the inequality. By definition of the infimum, we have that, for all $k \geq 1$,

$$\inf_{i\in\mathbb{N}*} AE(\mathcal{C}_i) = \frac{\sum_{i=1}^{k} |\mathcal{C}_i| \cdot \inf_{i\in\mathbb{N}*} AE(\mathcal{C}_i)}{\sum_{i=1}^{k} |\mathcal{C}_i|} \leq \frac{\sum_{i=1}^{k} |\mathcal{C}_i| \cdot AE(\mathcal{C}_i)}{\sum_{i=1}^{k} |\mathcal{C}_i|}.$$

Hence by taking the limit, we obtain

$$\inf_{i\in\mathbb{N}*} AE(\mathcal{C}_i) = \limsup_{k\to\infty} \left[\inf_{i\in\mathbb{N}*} AE(\mathcal{C}_i) \right] \leq \limsup_{k\to\infty} \left[\frac{\sum_{i=1}^{k} |\mathcal{C}_i| \cdot AE(\mathcal{C}_i)}{\sum_{i=1}^{k} |\mathcal{C}_i|} \right] = \overline{AE}(\pi).$$

This concludes our proof. □

C.3.3 One-player Games

We assume that the unique player is $\mathcal{P}_1$, hence that $S_2 = \emptyset$. The proofs are similar for the case where all states belong to $\mathcal{P}_2$ (i.e., $S_1 = \emptyset$). Similarly, we present our results for the $\overline{AE}$ variant, but they carry over to the $\underline{AE}$ one. Actually, since we show that we can restrict ourselves to *memoryless* strategies, all consistent outcomes will be periodic and thus both variants will be equal over those outcomes.

Memoryless determinacy. Intuitively, we use Lem. C.4 and Lem. C.5 to transform any arbitrary path in a simple *lasso path*, repeating a unique simple cycle, and yielding an at least as good AE, thus proving that any threshold achievable with memory can also be achieved without it.

Theorem C.6. *Memoryless strategies are sufficient to win one-player AE games.*

Proof. As a preliminary step, we look whether the graph contains a reachable strictly negative cycle, e.g., using the Bellman-Ford algorithm in $\mathcal{O}(|S| \cdot |T|)$-time. If so, then $\mathcal{P}_1$ can ensure a strictly negative mean-payoff, and by Point 1 of Lem. C.3, a memoryless strategy exists to make the average-energy be $-\infty$: such a strategy consists in reaching and repeating the negative simple cycle forever.

Now, assume that the graph contains no (reachable) strictly negative cycles. If the graph also contains no zero cycles, then the energy level necessarily diverges to $+\infty$, and the average-energy is $+\infty$ along any run. Indeed, we are in the case of Point 2 of Lem. C.3. Any strategy is optimal in that case: in particular, any memoryless strategy is.

For the rest of this proof, we assume that the graph contains no strictly negative cycles, but that it does contain zero cycles. An important consequence of this is that any subcycle of a zero cycle is a zero cycle. Now, consider an infinite path π with $\overline{AE}(\pi) \leq t$ (t being the threshold considered for the AE problem), and pick a state s that appears infinitely many times along π. The sequence of energy levels in the successive visits to s is nondecreasing (because there are no negative cycles), and for the average-energy to remain bounded along π, this sequence must converge to some value (Lem. C.2: $\overline{AE}(\pi) \geq \underline{TP}(\pi)$). Assume that the sequence is not constant: write ρ_0 for the prefix of π until the first visit to s, and π_2 for the "longest" suffix of π starting in s and along which the energy level when visiting s is constant. Finally, we let ρ_1 be the portion of π such that $\pi = \rho_0 \cdot \rho_1 \cdot \pi_2$. Applying Lem. C.4, we get

$$\begin{aligned}\overline{AE}(\rho_0 \cdot \pi_2) &= \mathrm{EL}(\rho_0) + \overline{AE}(\pi_2) \\ &\leq \mathrm{EL}(\rho_0) + \mathrm{EL}(\rho_1) + \overline{AE}(\pi_2) = \mathrm{EL}(\rho_0 \cdot \rho_1) + \overline{AE}(\pi_2) = \overline{AE}(\pi).\end{aligned}$$

Applying similar arguments, we can build an acyclic path ρ_0' based on ρ_0, if not already acyclic, such that

$$\overline{AE}(\rho_0' \cdot \pi_2) \leq \overline{AE}(\rho_0 \cdot \pi_2).$$

Now, by construction, π_2 can be decomposed into infinitely many simple zero cycles $(\mathcal{C}_i)_{i \in \mathbb{N}}$, starting and ending in s. This sequence of cycles satisfies all

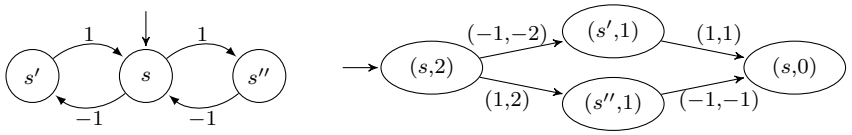
three conditions of Lem. C.5, and following Point 3, we get the existence of a simple cycle $\mathcal{C}$ such that

$$\overline{AE}(\rho'_0 \cdot \mathcal{C}^\omega) \leq \overline{AE}(\pi) \leq t.$$

The play $\pi' = \rho'_0 \cdot \mathcal{C}^\omega$ is a simple lasso path: this proves that memoryless strategies are sufficient to win one-player average-energy games. □

Polynomial-time algorithm. We now know the form of optimal memoryless strategies: an optimal lasso path $\pi = \rho \cdot \mathcal{C}^\omega$ w.r.t. the AE. We establish a polynomial-time algorithm to solve one-player AE games.

The crux of our algorithm consists in computing, for each state s, the best — w.r.t. the AE — *zero* cycle $\mathcal{C}_s$ starting and ending in s (if any). This is achieved through linear programming (LP) over expanded graphs. For each state s and length $k \in \{1, \ldots, |S|\}$, we compute the best cycle $\mathcal{C}_{s,k}$ by considering a graph (Fig. C.2) that models all cycles of length k from s and that uses $k+1$ levels and two-dimensional weights on edges of the form $(c, l \cdot c)$ where c is the weight in the original game and $l \in \{k, k-1, \ldots, 1\}$ is the level of the edge. In the LP, we look for cycles $\mathcal{C}_{s,k}$ of length k on s such that (a) the sum of weights in the first dimension is zero (thus $\mathcal{C}_{s,k}$ is a *zero* cycle), and (b) the sum in the second one is minimal. Fortunately, this sum is exactly equal to $AE(\mathcal{C}) \cdot k$ thanks to the l factors used in the weights of the expanded graph. Hence, we obtain the optimal cycle $\mathcal{C}_{s,k}$ (in polynomial time). Doing this $|S|$ times for each state s, we obtain for each of them the optimal cycle $\mathcal{C}_s$ (if one zero cycle exists). Then, by Lem. C.4, it remains to compute the least EL with which each state s can be reached using classical graph techniques (e.g., Bellman-Ford), and to pick the optimal combination to obtain an optimal memoryless strategy, in polynomial time.



(a) Original game. **(b)** Expanded graph for $k = 2$.

Figure C.2: The best cycle $\mathcal{C}_{s,2}$ is computed by looking for a path from $(s, 2)$ to $(s, 0)$ with sum zero in the first dimension (zero cycle) and minimal sum in the second dimension (minimal AE). Here, the cycle via s' is clearly better, with AE equal to $-1/2$ in contrast to $1/2$ via s''.

Theorem C.7. *The AE problem for one-player games is in* PTIME.

Proof. Let s_{init} be the initial state and $t \in \mathbb{Q}$ be the threshold. From Thm. C.6, we can restrict our search to *memoryless* strategies achieving average-energy less than or equal to t. As noted in the proof of Thm. C.6, if a strictly negative simple cycle exists and can be reached from s_{init}, then the answer to the AE problem is clearly Yes, as average-energy $-\infty$ is achievable. Checking if such a cycle exists and is reachable can be done in cubic time in the number of states (e.g., using Bellman-Ford to detect negative cycles).

Hence, we now assume that no negative cycle exists. The main part of our algorithm consists in computing, for each state s, the least average-energy that can be achieved along a simple *zero* cycle starting and ending in s (if any). Indeed, strictly positive cycles should be avoided as there is no negative cycle to counteract them. Applying Lem. C.4, it then remains to compute the least energy level with which each state s can be reached (simple paths are sufficient as there are no negative cycles), and to pick the optimal combination. Again, this last part can be solved by using classical graph algorithms in cubic time in $|S|$.

We now focus on computing the best zero cycle from a state s. This is achieved by enumerating the possible lengths, from 1 to $|S|$ (*simple* cycles suffice). For a fixed length k, we consider a new graph $\mathcal{G}_{s,k}$, made of $k+1$ copies of the original game $G(T)$. The states of $\mathcal{G}_{s,k}$ are pairs (u, l) with $u \in S$ and $0 \leq l \leq k$. The new graph is arranged in levels, indexed from $l = k$ for the top one to $l = 0$ for the bottom one: l represents the number of steps remaining to close the cycle of length k. For each edge (u, u') of $G(T)$, with $w(u, u') = c$, and for each $1 \leq l \leq k$, except if both $u' = s$ and $l < k$ (in order to rule out intermediary visits to s), there is an edge from (u, l) to $(u', l-1)$. This edge carries a pair of weights $(c, l \cdot c)$. Our aim is to find a path in this graph from (s, k) to $(s, 0)$ (hence this is a simple cycle of length k) such that the sum of the weights on the first dimension is zero (hence this is a zero cycle) and the sum on the second dimension is minimized (when divided by k, this sum is precisely the average-energy, if starting from energy level zero).

This problem can be expressed as a linear program, with variables $x_{u,u',l}$ for each edge $u \to u'$ and each $1 \leq l \leq k$. While they are not required to take integer values, these variables are intended to represent the number of times the edge from (u, l) to $(u', l-1)$ is taken along a "path" in $\mathcal{G}_{s,k}$. The linear program is as follows:

minimize $\sum x_{u,u',l} \cdot l \cdot w(u,u')$ subject to

1. $0 \leq x_{u,u',l} \leq 1$ for all $x_{u,u',l}$;
2. for all (u,l) with $1 \leq l \leq k-1$, $\sum_{u'} x_{u',u,l+1} = \sum_{u'} x_{u,u',l}$;
3. $\sum_{u'} x_{s,u',k} = \sum_{u'} x_{u',s,1} = 1$;
4. $\sum x_{u,u',l} \cdot w(u,u') = 0$;
5. $\sum x_{u,u',l} \geq 1$.

Condition (2) states that each state has the same amount of "incoming" and "outgoing" flow. Condition (3) expresses the fact that we start and end up in state s. Condition (4) encodes the fact that we are looking for zero cycles, and Condition (5) rules out the (possible) trivial solution where all variables are zero.

First observe that if this LP has no solution, then there is no zero cycle of length k from s. Now, assume it has a solution $(x^0_{u,u',l})$: this solution minimizes $\sum x_{u,u',l} \cdot l \cdot w(u,u')$. Consider a sequence of edges $s = u_k \to u_{k-1} \to \cdots \to u_1 \to u_0 = s$ for which $x_{u_l,u_{l-1},l} > 0$ for all l. The existence of such a sequence easily follows from Conditions (2) and (3). Assume that this is not a zero cycle. As there are no negative cycles, then this must be a positive cycle. But in order to fulfill Condition (4), we would need a negative cycle to compensate for this positive cycle, hence implying contradiction. We conclude that any sequence of consecutive edges as selected above is a zero cycle. Similarly, there cannot be a zero cycle of length k from s with better average-energy, as this would contradict the optimality of this solution. We thus have obtained an average-energy-optimal simple zero cycle of length k from s, in polynomial time. Indeed, the LP is polynomial in the size of $\mathcal{G}_{s,k}$, itself polynomial in the size of the original game: the expanded graph has its size bounded by $|S| \cdot (k+1)$ and all weights are bounded by $k \cdot W$ with $k \leq |S|$ and W the largest absolute weight in the original game.

As discussed above, this process can be repeated for each state s and each length k, $1 \leq k \leq |S|$, hence at most $|S|^2$ times. For each state, we select the best cycle among the $|S|$ possible ones (one for each length). Therefore, in polynomial time, we get a description of the best cycles w.r.t. the average-energy, for each $s \in S$. Clearly if no such cycle exists, then the answer to the AE problem is No, as all cycles are strictly positive and the average-energy of any play will be $+\infty$. If some exist, we can find an optimal strategy by picking the best combination between such a cycle from a state s and a corresponding prefix from s_{init} to s of minimal energy level. As presented before, this is achieved in polynomial time. Then the answer to the AE problem is Yes if and only if this optimal combination yields average-energy at most equal to t. This concludes our proof. □

C.3.4 Two-player Games

Memoryless determinacy. We now prove that memoryless strategies still suffice in two-player games. As discussed in Sect. C.3.2, most classical criteria do not apply. There is, however, one result that proves particularly useful. Consider any payoff function such that memoryless strategies suffice for *both one-player* versions ($S_1 = \emptyset$, resp. $S_2 = \emptyset$). In [69, Cor. 7], Gimbert and Zielonka establish that memoryless strategies also suffice in *two-player* games with the same payoff. Thanks to Thm. C.6, this entails the next theorem.

Theorem C.8. *Average-energy games are determined and both players have memoryless optimal strategies.*

Observe that this result is true for both variants of the average-energy payoff function, namely $\overline{AE}$ and $\underline{AE}$. When both players play optimally, they can restrict themselves to memoryless strategies and both variants thus coincide as mentioned earlier.

Solving average-energy games. Finally, consider the complexity of deciding the winner in a two-player AE game. By Thm. C.8, one can guess an optimal memoryless strategy for $\mathcal{P}_2$ and solve the remaining one-player game for $\mathcal{P}_1$, in polynomial time (by Thm. C.7). The converse is also true: one can guess the strategy of $\mathcal{P}_1$ and solve the remaining game where $S_1 = \emptyset$ in polynomial time. Thus, we obtain the following result.

Theorem C.9. *The AE problem for two-player games is in* $\mathsf{NP} \cap \mathsf{coNP}$.

We complete our study by proving that MP games can be encoded into AE ones in polynomial time. The former are known to be in $\mathsf{NP} \cap \mathsf{coNP}$ but whether they belong to PTIME is a long-standing open question (e.g., [126, 79, 26, 33]). Hence, w.r.t. current knowledge, the $\mathsf{NP} \cap \mathsf{coNP}$-membership of the AE problem can be considered optimal. The key of the construction is to double each edge of the original game and modify the weight function such that each pair of successive edges corresponding to such a doubled edge now has a total energy level of zero, and an average-energy that is exactly equal to the weight of the original edge. Then we apply decomposition techniques as in Lem. C.5 to establish the equivalence.

Theorem C.10. *Mean-payoff games can be reduced to average-energy games in polynomial time.*

Proof. Let $G = (S_1, S_2, E, w)$ be a game, and $t \in \mathbb{Q}$ be the threshold for the

mean-payoff problem. From G, we build another game $G' = (S'_1, S'_2, E', w')$ such that

- $S'_1 = S_1 \cup E$ and $S'_2 = S_2$;
- E' contains two types of edges:
  - $(s, e) \in E'$ iff there exists s' such that $e = (s, s') \in E$. Then $w'(s, e) = 2 \cdot w(e)$.
  - $(e, s') \in E'$ for any $e = (s, s') \in E$. Then $w'(e, s') = -2 \cdot w(e)$.

We claim that $\mathcal{P}_1$ has a strategy ensuring objective *MeanPayoff*(t) in G if and only if the answer for the AE problem in G' is Yes for the same threshold t. A similar construction is used in [17].

With a prefix $\rho = (s_i)_{i \leq n}$ in G, we can associate a prefix $\rho' = (s'_i)_{i \leq 2n}$ in G' as follows: for all $k \leq n$, $s'_{2k} = s_k$, and for all $k < n$, $s'_{2k+1} = (s_k, s_{k+1})$. The mean-payoff along ρ then equals the average energy along ρ' (assuming initial energy 0 for ρ'). Indeed, applying the same decomposition arguments as for Lem. C.5 and by definition of the weight function w', we have that

$$\begin{aligned} AE(\rho') &= \frac{1}{n} \sum_{i=0}^{n-1} \frac{2 \cdot w'(s_i, (s_i, s_{i+1})) + w'((s_i, s_{i+1}), s_{i+1})}{2} \\ &= \frac{1}{n} \sum_{i=0}^{n-1} \frac{4 \cdot w(s_i, s_{i+1}) - 2 \cdot w(s_i, s_{i+1})}{2} = \frac{1}{n} \sum_{i=0}^{n-1} w(s_i, s_{i+1}) = \mathsf{MeanPayoff}(\rho). \end{aligned}$$

Conversely, with a prefix $\rho' = (s'_i)_{i \leq 2n}$ in G' starting and ending in a state in $S_1 \cup S_2$, we can associate a prefix $\rho = (s_i)_{i \leq n}$ in G such that $s_k = s'_{2k}$ for all $k \leq n$. Again, assuming the initial energy is zero in ρ', the average energy along ρ' equals the mean payoff along ρ.

Now, assume that $\mathcal{P}_1$ has a winning strategy σ in G from some state $s \in S_1 \cup S_2$, achieving mean-payoff less than or equal to t. Consider the strategy σ' for G' defined as $\sigma'(\rho') = \sigma(\rho)$ if ρ' ends in S_1. If ρ' ends in a T-state of the form (s, s'), then we let $\sigma'(\rho') = s'$, which is the only possible outgoing edge. We see that the outcomes of σ' correspond to the outcomes of σ, so that, assuming that the initial energy level is zero, σ' enforces that the average-energy is below t for any infinite outcome. Conversely, given a strategy σ' for G' whose outcomes have average-energy below t, the strategy defined by $\sigma(\rho) = \sigma'(\rho')$ for all finite paths ρ in G secures a mean-payoff below t. Observe that the equivalence holds both between $\overline{AE}$ and $\overline{MP}$, and between $\underline{AE}$ and $\underline{MP}$. Indeed, we have seen that for both MP and AE games, memoryless strategies suffice and decision problems for both variants coincide. □

C.4 Average-Energy with Lower- and Upper-Bounded Energy

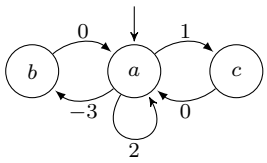
We extend the *AE* framework with constraints on the running energy level of the system. Such constraints are natural in many applications where the energy capacity is bounded (e.g., fuel tank, battery charge). We first study the case where the energy is subject to *both* a lower bound (here, zero) *and* an upper bound ($U \in \mathbb{N}$). We study the problem for the *fixed initial energy level* $c_{\mathsf{init}} \coloneqq 0$. In this case, the range of acceptable energy levels is by definition constrained to the interval $[0, U]$. Our approach benefits from this: we solve the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem by considering an *AE* problem (and subsequently, an *MP* problem) over an expanded game that explicitly accounts for the lower and upper bounds on the energy.

Formally, we want to decide if $\mathcal{P}_1$ can ensure a *sufficiently low AE* while keeping the EL within the allowed range.

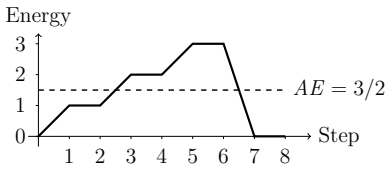
Problem C.2 ($\mathsf{AvgEnergy}_{\mathsf{LU}}$)**.** *Given a game $G(T)$, an initial state s_{init}, an upper bound $U \in \mathbb{N}$, and a threshold $t \in \mathbb{Q}$, decide if $\mathcal{P}_1$ has a winning strategy $\sigma_1 \in \Sigma_1$ for the objective $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t)$.*

Again, we present results for the supremum variant $\overline{AE}$ but they also hold for the infimum one $\underline{AE}$.

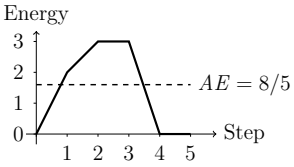
Illustration. Consider the one-player game in Fig. C.3. The energy constraints force $\mathcal{P}_1$ to keep the energy in [0, 3] at all times. Hence, only three strategies can be followed safely, respectively inducing plays π_1, π_2 and π_3. Due to the bounds on energy, it is natural that strategies need to alternate between both a positive and a negative cycle to satisfy objective $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0)$ (since no simple zero cycle exists). It is yet interesting that to play optimally (play π_3), $\mathcal{P}_1$ actually has to use *both* positive cycles, and in the *appropriate order* (compare plays π_2 and π_3).
This type of alternating behavior is more intricate than for other classical conjunctions of objectives. Consider for example energy parity [38] or multi-dimensional energy games [36, 121]. It is usually necessary to use different cycles in such games: intuitively, one needs one "good" cycle for each dimension and one for the parity objective, and a winning strategy needs to alternate between those cycles. However, there is no need to use *two* different cycles that are "good" w.r.t. the same part of the objective. In the case of $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games, we see that it is sometimes necessary to use two (or more) different cycles even though they impact the sum of weights in the same direction (e.g.,



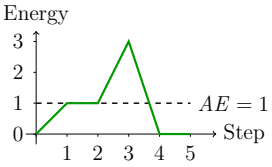
(a) One-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game.



(b) Play $\pi_1 = (acacacab)^\omega$.



(c) Play $\pi_2 = (aacab)^\omega$.



(d) Play $\pi_3 = (acaab)^\omega$.

Figure C.3: Example of a one-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game ($U = 3$) and the evolution of energy under different strategies that maintain it within $[0, 3]$ at all times. The minimal average-energy is obtained with play π_3: alternating in order between the $+1$, $+2$ and -3 cycles.

several positive cycles). This gives a hint of the complexity of $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games.

C.4.1 Pseudo-polynomial Algorithm and Complexity Bounds

We first reduce the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem to the AE problem over a *pseudo-polynomial expanded game*, i.e., polynomial in the size of the original $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game and in $U \in \mathbb{N}$. By Thm. C.9 and Thm. C.7, this reduction induces $\mathsf{NEXPTIME} \cap \mathsf{coNEXPTIME}$-membership of the two-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem, and $\mathsf{EXPTIME}$-membership of the one-player one. We improve the complexity for two-player games by further reducing the AE game to an $\mathsf{MeanPayoff}$ game: this yields $\mathsf{EXPTIME}$-membership, which is optimal (Thm. C.13). We also improve the one-player case by observing that a witness lasso path in the $\mathsf{MeanPayoff}$ game can be built on-the-fly, and the mean-payoff of this path can be computed using only polynomial space in the original game, hence we end up with PSPACE-membership which we also prove optimal in Thm. C.13.

Observe that if U is encoded in unary or if U is polynomial in the size of the original game, the complexity of the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem collapses to $\mathsf{NP} \cap \mathsf{coNP}$ for two-player games and to PTIME for one-player games thanks to

our reduction to an AE problem and the results of Thm. C.9 and Thm. C.7.

The reductions. Given a game $G = (S_1, S_2, E, w)$, an initial state s_{init}, an upper bound $U \in \mathbb{N}$, and a threshold $t \in \mathbb{Q}$, we reduce the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem to an AE problem as follows. If at any point along a play, the energy drops below zero or exceeds U, the play will be losing for the $Energy_{LU}(U, c_{\mathsf{init}} \coloneqq 0)$ objective, hence also for its conjunction with the AE one. So we build a new game G' over the state space $(S \times \{0, 1, \ldots, U\}) \cup \{\mathsf{sink}\}$. The idea is to include the energy level within the state labels, with sink as an absorbing state reached only when the energy constraint is breached. We now consider the AE problem for threshold t on G'. By putting a self-loop of weight 1 on sink, we ensure that if the energy constraint is not guaranteed in G, the answer to the AE problem in G' will be No as the average-energy will be infinite due to reaching this positive loop and repeating it forever. Hence, we show that the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ objective can be won in G if and only if the AE one can be won in G' (thus avoiding the sink state). The result of the reduction for the game in Fig. C.3a is presented in Fig. C.4.

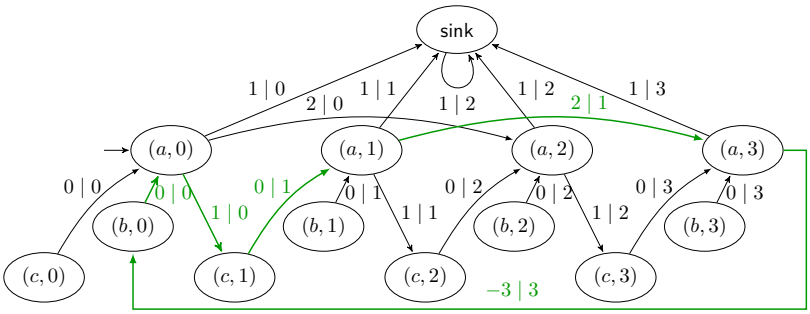


Figure C.4: Reduction from the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game in Fig. C.3a to an AE game and further reduction to an $\mathsf{MeanPayoff}$ game over the same expanded graph. For the sake of succinctness, the weights are written as $c \mid c'$ with c the weight used in the AE game and c' the one used in the $\mathsf{MeanPayoff}$ game. We use the upper bound $U = 3$ and the average-energy threshold $t = 1$ (the optimal value in this case). The optimal play $\pi_3 = (acaab)^\omega$ of the original game corresponds to an optimal memoryless play in the expanded graph.

Lemma C.11. *The* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *problem over a game* $G = (S_1, S_2, E, w)$*, with an initial state* s_{init}*, an upper bound* $U \in \mathbb{N}$*, and a threshold* $t \in \mathbb{Q}$*, is reducible to an* AE *problem for the same threshold* $t \in \mathbb{Q}$ *over a game* $G' = (S'_1, S'_2, E', w')$ *such that* $|S'| = (U+1) \cdot |S| + 1$ *and* $W' = \max\{\min\{W, U\}, 1\}$*, i.e., the largest absolute weight in* G' *is at most the same as in* G*, or equal to constant* 1*.*

Proof. Consider the game $G = (S_1, S_2, E, w)$, with initial state s_{init}, upper

bound $U \in \mathbb{N}$ and threshold $t \in \mathbb{Q}$. We define the expanded game $G' = (S'_1, S'_2, E', w')$ as follows.

- $S'_1 = (S_1 \times \{0, 1, \dots, U\}) \cup \{\mathsf{sink}\}$.
- $S'_2 = S_2 \times \{0, 1, \dots, U\}$.
- For all $(u, v) \in E$, $(u, c) \in S'$, we have that:
  1. if $d = c + w(u, v) \in [0, U]$, then $e = \Big((u, c), (v, d)\Big) \in E'$ and $w'(e) = w(u, v)$,
  2. else $e = \Big((u, c), \mathsf{sink}\Big) \in E'$ and $w'(e) = 1$.
- $(\mathsf{sink}, \mathsf{sink}) \in E'$ and $w(\mathsf{sink}, \mathsf{sink}) = 1$.

The game G' starts in state $(s_{\mathsf{init}}, 0)$ and edges are built naturally to reflect the changes in the energy level. Whenever the energy drops below zero or exceeds U, we redirect the edge to sink, where a self-loop of weight 1 is repeated forever.

We claim that $\mathcal{P}_1$ has a winning strategy σ_1 for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ objective in G if and only if he has a winning strategy σ'_1 for the AE objective in G', for the very same average-energy threshold t.

First, consider the left-to-right implication. Assume σ_1 is winning for objective $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} := 0) \cap \mathit{AvgEnergy}(t)$ in G. The very same strategy can be followed in G', ignoring the additional information on the energy in the state labels. Precisely, for any prefix $\rho' = (s_0, c_0)(s_1, c_1) \dots (s_n, c_n)$ in G', we define $\sigma'_1(\rho') = (s, c)$ where $s = \sigma_1(\rho)$ for $\rho = s_0 s_1 \dots s_n$ and $c = c_n + w(s_n, s)$. Obviously, playing this strategy ensures that the special state sink is never reached, as otherwise it would not be winning for $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0)$ in G, by construction of G'. Since all weights are identical in both games except on edges entering the sink state, we have that any consistent outcome π' of σ'_1 in G' corresponds to a consistent outcome π of σ_1 in G such that $\overline{AE}(\pi') = \overline{AE}(\pi)$, and conversely. Therefore, σ' is clearly winning for objective $\mathit{AvgEnergy}(t)$ in G'.

Second, consider the right-to-left implication. Assume σ'_1 is winning for objective $\mathit{AvgEnergy}(t)$ in G'. Then this strategy ensures that sink is avoided forever. Otherwise, there would exist a consistent outcome π' reaching sink, and such that $\overline{AE}(\pi') = \infty > t$ because of the strictly positive self-loop. Thus the strategy would not be winning. Hence by construction of G', this strategy trivially ensures $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0)$ in G'. From σ'_1, we build a strategy σ_1 in G in the natural way, potentially integrating the information on the energy within the memory of σ_1. Again, there is a bijection between plays avoiding

sink in G' and plays in G, such that σ_1 is winning for $Energy_{LU}(U, c_{\mathsf{init}} := 0) \cap AvgEnergy(t)$ in G.

Hence we have shown the claimed reduction. For the sake of completeness, observe that the reduction holds both for $\overline{AE}$ and $\underline{AE}$ variants of the average-energy. It remains to discuss the size of the expanded game. Observe that $|S'| = (U+1) \cdot |S| + 1$. Furthermore, if W is the largest absolute weight in G, then $W' = \max\{\min\{W, U\}, 1\}$ is the largest one in G'. Indeed, W' is upper-bounded by U by construction (as all edges of absolute weight larger than U can be redirected directly to sink) and it is lower-bounded by 1 due to edges leading to sink. So the state space of G' is polynomial in the state space of G and in the *value* of the upper bound U, while its weights are bounded by either the largest weight W, the upper bound U or constant 1. □

We now show that the AE game G' can be further reduced to an MeanPayoff game G'' by modifying the weight structure of the graph. Essentially, all edges leaving a state (s, c) of G' are given weight c in G'', i.e., the current energy level, and the self-loop on sink is given weight $(\lceil t \rceil + 1)$. This modification is depicted in Fig. C.4. We claim that the AE problem for threshold $t \in \mathbb{Q}$ in G' is equivalent to the MP problem for the same threshold in G''. Indeed, we show that with our change of weight function, reaching sink implies losing, both in G' for AE and in G'' for MP, and all plays that *do not* reach sink have the same value for their average-energy in G' as for their mean-payoff in G''.

Lemma C.12. *The AE problem over the game $G' = (S'_1, S'_2, E', w')$ defined in Lem. C.11 is reducible to an MP problem for the same threshold $t \in \mathbb{Q}$ over a game $G'' = (S'_1, S'_2, E', w'')$ sharing the same state space but with largest absolute weight $W'' = \max\{U, \lceil t \rceil + 1\}$, where U is the energy upper bound of the original* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *problem.*

Proof. Let $G' = (S'_1, S'_2, E', w')$ be the game defined in Lem. C.11, as a reduction from the original game G for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem with upper bound $U \in \mathbb{N}$ and average-energy threshold $t \in \mathbb{Q}$. We now build the game $G'' = (S'_1, S'_2, E', w'')$ by simply modifying the weight function of G'. The changes are as follows:

- For all edge $e = ((s, c), (s', c')) \in E'$, its weight in G' is $w'(e) = c' - c$ and we now set it to $w''(e) = c$ in G''. Recall that by construction of G', the value c represents the current energy level for any prefix ending in (s, c). This is the value we now use for the outgoing edge. Also, this value is constrained in $[0, U]$ by definition of G'.
- For all edge $e = ((s, c), \mathsf{sink}) \in E'$, its weight in G' is $w'(e) = 1$ and we

now set it to $w''(e) = c$ in G'' for the sake of consistency (the actual value over this type of edges will not matter eventually).

- For the self-loop $e = (\mathsf{sink}, \mathsf{sink}) \in E'$, its weight in G' is $w'(e) = 1$ and we now set it to $w''(e) = \lceil t \rceil + 1$ in G''. That is, reaching sink will imply a mean-payoff higher than the threshold.

Before proving the claim, we show that for all plays $\pi \in \mathit{Plays}(G') = \mathit{Plays}(G'')$ that *do not* reach sink, we have that $\overline{AE}_{G'}(\pi) = \overline{MP}_{G''}(\pi)$, where the subscript naturally refers to the change of weight function. Let

$$\pi = s'_0 s'_1 s'_2 \ldots = (s_0, c_0)(s_1, c_1)(s_2, c_2)\ldots$$

be such a play, where for all $i \geq 0$, $s'_i \in S'$ and $(s_i, c_i) \in S \times [0, U] \cap \mathbb{N}$ is its corresponding label. By definition of G'', we have that,

$$\forall\, n \geq 0, \quad w''(s'_n, s'_{n+1}) = c_n = \mathrm{EL}_{G'}(\pi(n)).$$

Hence by definition of the mean-payoff and the average-energy,

$$\begin{aligned} \overline{MP}_{G''}(\pi) &= \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} w''(s'_i, s'_{i+1}) \\ &= \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathrm{EL}_{G'}(\pi(i)) = \overline{AE}_{G'}(\pi). \end{aligned} \tag{C.4}$$

For the sake of completeness, observe that this equality does not hold for plays reaching sink, as they have infinite average-energy in G' but finite mean-payoff in G''.

We proceed by proving the claim that $\mathcal{P}_1$ has a winning strategy σ'_1 for the AE objective in G' if and only if he has a winning strategy σ''_1 for the $\mathsf{MeanPayoff}$ objective in G'', for the very same threshold t.

First, consider the left-to-right implication. Assume σ'_1 is winning for objective $\mathit{AvgEnergy}(t)$ in G'. We apply the same strategy in G'' straightforwardly as the underlying graph is not modified. Since this strategy is winning for the AE objective in G', it necessarily avoids sink both in G' and G'' (as otherwise the AE would be infinite). Hence by Eq. (C.4), we have that σ'_1 is also winning for $\mathit{MeanPayoff}(t)$ in G''.

Second, consider the right-to-left implication. Assume σ''_1 is winning for objective $\mathit{MeanPayoff}(t)$ in G''. Since the self-loop on sink has weight $\lceil t \rceil + 1$, it is necessary that σ''_1 never reaches sink otherwise it would not be winning. Hence we apply the same strategy in G' and by Eq. (C.4), we have that σ''_1 is also winning for $\mathit{AvgEnergy}(t)$ in G'.

This proves correctness of the reduction. The same reasoning can be followed for $\underline{AE}$ (thus using $\underline{MP}$) instead of $\overline{AE}$. We end by discussing the size of G''. Clearly, the state space S'' is identical to S', hence $|S''| = (U+1) \cdot |S| + 1$. However, the largest absolute weight in G'' is $W'' = \max\{U, \lceil t \rceil + 1\}$. Indeed, the self-loop on sink has weight $(\lceil t \rceil + 1)$ and all other edges have weight bounded by the energy upper bound U by construction. □

Illustration. Consider the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game G depicted in Fig. C.3a. We have seen that the optimal strategy is $\pi_3 = (acaab)^\omega$. Now consider the reduction to the AE game, and further to the MeanPayoff game, depicted in Fig. C.4. The optimal (memoryless) strategy in both the AE game G' and the MP game G'' is to create the play $\pi' = ((a,0)(c,1)(a,1)(a,3)(b,0))^\omega$, which corresponds to the optimal play π_3 in the original game. It can be checked that $\overline{AE}_G(\pi_3) = \overline{AE}_{G'}(\pi') = \overline{MP}_{G''}(\pi')$.

Complexity. The reduction from the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game to the AE one induces a pseudo-polynomial blow-up in the number of states. Thanks to the second reduction and the use of a pseudo-polynomial algorithm for the MeanPayoff game [126, 26], we get EXPTIME-membership, which is optimal for two-player games thanks to the lower bound proved for $\mathsf{Energy}_{\mathsf{LU}}$ [18]. The complexity is reduced when the bound U is given in unary or is polynomial in the size of the game, matching the one obtained for AE games without energy constraints.

For the one-player case, we also use the reduction to an MeanPayoff game. By [54], optimal memoryless strategies exist, hence it suffices to non-deterministically build a simple lasso path in G'', and to check that it satisfies the mean-payoff constraint. It can be done using only polynomial space through on-the-fly computation.

Theorem C.13. *The* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *problem is EXPTIME-complete for two-player games and PSPACE-complete for one-player games. If the upper bound* $U \in \mathbb{N}$ *is polynomial in the size of the game or encoded in unary, the* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *problem collapses to* $\mathsf{NP} \cap \mathsf{coNP}$ *and PTIME for two-player and one-player games, respectively.*

Proof. Let $G = (S_1, S_2, E, w)$ be the original $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game, $W \in \mathbb{N}$ its largest absolute weight, $U \in \mathbb{N}$ the upper bound for energy and $t \in \mathbb{Q}$ the threshold for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem. By Lem. C.11, this $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem is reducible to an AE problem for the same threshold t over a game $G' = (S'_1, S'_2, E', w')$ such that $|S'| = (U+1) \cdot |S| + 1$ and $W' =$

$\max\{\min\{W, U\}, 1\}$. By Lem. C.12, the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem can be further reduced to an MP problem for the same threshold t over a game $G'' = (S''_1, S''_2, E', w'')$ sharing the same state space as G' but with largest absolute weight $W'' = \max\{U, \lceil t \rceil + 1\}$. We start by proving the complexity upper bounds.

First, consider the one-player case. Combining Thm. C.7 and the reduction to an AE game, we obtain that one-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games can be solved in pseudo-polynomial time, i.e., polynomial in $|S|$ but also in the value of U (hence exponential in the size of its binary encoding). This both gives EXPTIME-membership of one-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games with arbitrary upper bounds, and PTIME-membership of the same games with polynomial or unary upper bounds. For arbitrary bounds, we improve the complexity from EXPTIME to PSPACE. To do so, we consider the further reduction to an MeanPayoff game, but we *do not* completely build the MeanPayoff game G'' which is known to be of exponential size. Instead, we build non-deterministically a witness lasso path (thanks to memoryless determinacy [54], they are sufficient) and check on-the-fly that the path is winning or not, using only polynomial space. Recall that we consider a game G'' such that $S''_2 = \emptyset$. Our non-deterministic algorithm answers Yes if $\mathcal{P}_1$ has a winning strategy in G'' (and hence in G thanks to Lem. C.11 and Lem. C.12), No otherwise, and is as follows:

1. Guess a state $s'_r \in S'_1 = (S_1 \times \{0, 1, \dots, U\}) \cup \{\mathsf{sink}\}$ that will be the starting (and ending) state of the cycling part of the lasso path. For the following, we assume that $s'_r \neq \mathsf{sink}$ otherwise the lasso path that we are trying to build is clearly losing (see proof of Lem. C.12) and the algorithm answers No. Thus, store state $s'_r = (s_r, m)$ for some $m \in \{0, \dots, U\}$.

2. Check that s'_r is reachable from the initial state $(s_{\mathsf{init}}, 0)$. This can be done in NLOGSPACE w.r.t. the size of G'' (see e.g., [118]), hence NPSPACE w.r.t. the original problem. If it is not, then the answer is No.

3. Build step by step[2] a lasso path by constructing a simple cycle in G'' starting in s'_r. This construction is non-deterministic: if at any point, the sink state is reached, the algorithm returns No. The construction stops as soon as s'_r is reached, or after $|S'| + 1$ steps if s'_r is not reached: in the latter case, the answer is also No (after $|S'| + 1$ steps, we know for certain that a cycle was created hence our lasso path is complete). While constructing the cycle, we make on-the-fly computations: at each step, the next state is chosen non-deterministically and the only information

[2]Observe that given a state in G'', it is indeed possible to build any neighboring state using only E and w from the original game: one can effectively build the graph G'' on-the-fly.

that is stored — except from state s'_r used to determine the end of the cycle — is the number of steps from leaving s'_r, and the sum of the weights seen along the cycle.

4. Assume s'_r is reached (otherwise we have seen that the answer is No). Let $s'_0 s'_1 \dots s'_l$ be the sequence of states visited along the construction, with $s'_0 = s'_l = s'_r$. We have stored the length l and the sum of weights

$$\gamma = \sum_{i=0}^{l-1} w''(s'_i, s'_{i+1}).$$

Now, we check if $\frac{\gamma}{l} \leq t$: this quantity is the mean-payoff of the lasso path we have constructed. If yes, then the answer is Yes, thanks to Lem. C.11 and Lem. C.12: the lasso path describes a winning strategy. Otherwise, the answer is No as this lasso path represents a losing strategy, by the same lemmas.

The correctness of this algorithm is guaranteed by Lem. C.11 and Lem. C.12. It remains to argue that it only uses polynomial space in the original $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem. Observe that our on-the-fly computations only need to record the state s'_r, the current state, the current length and the current sum. We have that both states belong to $S_1 \times \{0, 1, \dots, U\}$, that $l < |S'| = (U+1) \cdot |S| + 1$ and that the sum is bounded by $l \cdot W'' = l \cdot \max\{U, \lceil t \rceil + 1\}$. Hence, encoding those values only requires a polynomial number of bits w.r.t. the input of the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem (i.e., logarithmic in the upper bound U, the largest weight W and the threshold t). This proves that our algorithm lies in NPSPACE, and by Savitch's theorem [118] we know that NPSPACE = PSPACE: hence we proved the upper bound for the one-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem.

Second, consider two-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games. In this case, we solve the MP problem over G'' using a pseudo-polynomial algorithm such as the one presented in [26], whose complexity is $\mathcal{O}(|S^*|^3 \cdot W^*)$ for a game with $|S^*|$ states and largest absolute weight $W^* \in \mathbb{N}$. Therefore, the complexity of solving the original $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem is

$$\mathcal{O}\left(|S'|^3 \cdot W''\right) = \mathcal{O}\left(\left((U+1) \cdot |S| + 1\right)^3 \cdot \max\{U, \lceil t \rceil + 1\}\right),$$

which is clearly pseudo-polynomial. Hence we obtain EXPTIME-membership for two-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games. If the upper bound $U \in \mathbb{N}$ is polynomial in the size of the game or encoded in unary, it is sufficient to solve the polynomially-larger AE game G' using Thm. C.9 to obtain NP ∩ coNP-membership.

Now consider lower bounds. The $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem trivially encompasses the lower- and upper-bounded energy problem $\mathsf{Energy}_{\mathsf{LU}}$, i.e., the $\mathsf{AvgEnergy}_{\mathsf{LU}}$

without consideration of the average-energy. Indeed, consider a game G with an objective $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0)$, for some $U \in \mathbb{N}$. Assume $\mathcal{P}_1$ has a winning strategy for this objective. Then this strategy ensures that along any consistent outcome π, the running energy at any point is at most equal to U. By definition, this implies that $\underline{AE}(\pi) \leq \overline{AE}(\pi) \leq U$. Hence this strategy is also winning for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ objective written as the conjunction $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t \coloneqq U)$. The converse is also trivially true. Ergo, any lower bound on the complexity of the $\mathsf{Energy}_{\mathsf{LU}}$ problem also holds for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ one. The EXPTIME-hardness of the two-player $\mathsf{Energy}_{\mathsf{LU}}$ problem was proved in [18], the PSPACE-hardness of the one-player version was proved in [61] (in the equivalent setting of reachability in bounded one-counter automata). Note that those results clearly rely on having an upper bound U larger than polynomial (w.r.t. the size of the game) and encoded in binary, as we have already shown that in the opposite case the complexity of the problem is reduced.

Finally, observe that the same reduction and complexities also hold if we use $\underline{AE}$ instead of $\overline{AE}$ to define the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem. This concludes our proof. □

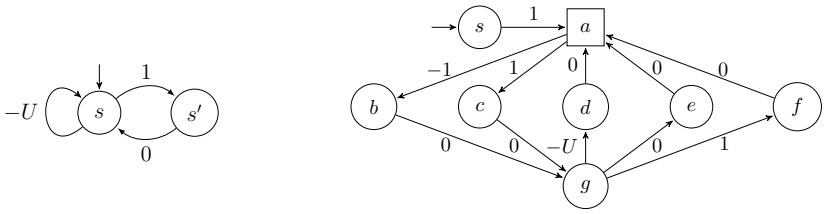
Remark C.14. *One could argue that the reduction from AE games to MP games presented in Lem. C.12 could be used to solve AE games without resorting to the specific analysis of Sect. C.3. Indeed, in the case where the mean-payoff value is zero, any memoryless strategy (which we know to suffice) that is winning should only create zero-cycles: the energy can be constrained in the range* $[-2 \cdot |S| \cdot W, 2 \cdot |S| \cdot W]$ *along any winning play. However, applying a pseudo-polynomial MP algorithm on this new game would only grant EXPTIME-membership for AE games (because of the polynomial dependency on* W*), in contrast to the* $\mathsf{NP} \cap \mathsf{coNP}$ *and PTIME results obtained with the refined analysis for two-player and one-player AE games respectively.*

C.4.2 Memory Requirements

We prove pseudo-polynomial lower and upper bounds on memory for the two players in $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games. The upper bound follows from the reduction to a pseudo-polynomial AE game and the memoryless determinacy of AE games proved in Thm. C.8. Observe that winning strategies obtained via our reductions have a natural form: they are memoryless w.r.t. configurations (s, c) denoting the current state and the current energy level. As noted before, when the upper bound on energy $U \in \mathbb{N}$ is polynomial or given in unary, the expanded game is only polynomial in size, and the memory needs are also

reduced.

The lower bound can be witnessed in two families of games asking for strategies using memory polynomial in the energy upper bound $U \in \mathbb{N}$ to be won by $\mathcal{P}_1$ (Fig. C.5a) or $\mathcal{P}_2$ (Fig. C.5b) respectively. It is interesting to observe that those families already ask for such memory when considering the simpler $\mathsf{Energy}_{\mathsf{LU}}$ objective (i.e., bounded energy only). Sufficiency of pseudo-polynomial memory for $\mathsf{Energy}_{\mathsf{LU}}$ games follows from [18] but to the best of our knowledge, it was not proved in the literature that such memory is also necessary.



(a) $\mathcal{P}_1$ needs to take U times (s, s') before taking (s, s) once and repeating.

(b) $\mathcal{P}_2$ needs to increase the energy up to U using (a, c) to force $\mathcal{P}_1$ to take (g, d) then make him lose by taking (a, b).

Figure C.5: Families of games witnessing the need for pseudo-polynomial-memory strategies for $\mathsf{Energy}_{\mathsf{LU}}$ (and $\mathsf{AvgEnergy}_{\mathsf{LU}}$) objectives. The goal of $\mathcal{P}_1$ is to keep the energy in $[0, U]$ at all times, for $U \in \mathbb{N}$. The left game is won by $\mathcal{P}_1$ and the right one by $\mathcal{P}_2$ but both require memory polynomial in the *value* U to be won.

Theorem C.15. *Pseudo-polynomial-memory strategies are both sufficient and necessary to win in* $\mathsf{Energy}_{\mathsf{LU}}$ *and* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *games with arbitrary energy upper bound* $U \in \mathbb{N}$*, for both players. Polynomial memory suffices when* U *is polynomial in the size of the game or encoded in unary.*

Proof. We first prove the *upper bound on memory.* The expanded game G' built in the reduction from the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ to the AE problem (Lem. C.11) has a state space of size $|S'| = (U + 1) \cdot |S| + 1$, over which memoryless strategies suffice, by Thm. C.8. Thus, winning for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ objective only requires memory that is polynomial in the original number of states and the upper bound value $U \in \mathbb{N}$. The same reduction holds for $\mathsf{Energy}_{\mathsf{LU}}$ games with an even simpler *safety* objective (never reaching sink) instead of the AE one (or equivalently with the AE objective for threshold $t = U$). Thus, with regard to the *binary encoding* of U, strategies require exponential memory in general. For the special cases of unary encoding or polynomially bounded value U, polynomial memory suffices. Note that as usual, these arguments are true for both the $\overline{AE}$ and the $\underline{AE}$ versions of the objective.

We now discuss the two families of games witnessing that pseudo-polynomial memory is also a *lower bound* for both players.

First, consider the one-player game depicted in Fig. C.5a and parametrized by the value $U \in \mathbb{N}$. Assume the objective is $\mathsf{Energy}_{\mathsf{LU}}$, asking for the energy to remain within $[0, U]$ at all times. Recall that the initial energy level is fixed to $c_{\mathsf{init}} \coloneqq 0$. It is easy to see that there is only one acceptable strategy for $\mathcal{P}_1$: playing (s, s') exactly U times, then playing the self-loop (s, s) once, and repeating this forever. Indeed, any other strategy eventually leads the energy outside the allowed range. Hence, to win this game, $\mathcal{P}_1$ needs a strategy described by a Moore machine whose memory contains at least $(U + 1)$ states. This proves that pseudo-polynomial memory is required for $\mathcal{P}_1$ in $\mathsf{Energy}_{\mathsf{LU}}$ games. Furthermore, the same argument can be applied on this game with objective $\mathsf{AvgEnergy}_{\mathsf{LU}}$ by considering the average-energy threshold $t \coloneqq U$ which is trivially ensured by strategies satisfying the $\mathsf{Energy}_{\mathsf{LU}}$ objective.

Second, consider the two-player[3] $\mathsf{Energy}_{\mathsf{LU}}$ game depicted in Fig. C.5b. Again this game is parametrized by the energy upper bound $U \in \mathbb{N}$ and the initial energy level is fixed to $c_{\mathsf{init}} \coloneqq 0$. This game can be won by $\mathcal{P}_2$ using the following strategy: if the energy level is in $[1, U]$, play (a, c), otherwise play (a, b). Note that this strategy again requires at least $(U + 1)$ states of memory in its Moore machine (to keep track of the energy level).

This strategy is indeed winning. Observe that $\mathcal{P}_1$ can only decrease the energy by using edge (g, d) of weight $-U$, and this edge can only be used safely if the energy level is exactly U. In addition, the energy is bound to reach or exceed U eventually (as it will increase by 1 or 2 between each visit of a). If it exceeds U, then $\mathcal{P}_2$ wins directly. Otherwise, assume that the energy is U when the game is in state g. If $\mathcal{P}_1$ plays (g, f), he loses (the energy reaches $U + 1$). If he plays (g, e), $\mathcal{P}_2$ wins by playing (a, c) (the energy also reaches $U + 1$). And if $\mathcal{P}_1$ plays (g, d), $\mathcal{P}_2$ wins by playing (a, b) (the energy reaches -1). Hence, $\mathcal{P}_2$ wins the game against all strategies of $\mathcal{P}_1$.

Now, observe that $\mathcal{P}_2$ *cannot* win if he uses a strategy with less memory states in its Moore machine. Indeed, any such strategy cannot keep track of all the energy levels between 0 and U and play (a, c) a sufficient number of times in a row before switching to the appropriate choice (depending on the energy being 0 or U). Therefore, if $\mathcal{P}_2$ uses such a strategy, $\mathcal{P}_1$ can maintain the energy in the allowed range by simply reacting to edge (a, b) with (g, f) and to edge (a, c) by choosing between (g, d) (if the energy is U) and (g, e) (otherwise). Such

[3] In $\mathsf{Energy}_{\mathsf{LU}}$ games with only $\mathcal{P}_2$ (i.e., $S_1 = \emptyset$), $\mathcal{P}_2$ does not need memory to play as he can pick beforehand which of the energy bounds (lower or upper) he will transgress, and then do so with a memoryless strategy.

choices are safe for $\mathcal{P}_1$ as the strategy of $\mathcal{P}_2$ does not have enough memory to distinguish the resulting energy levels from the intermediate ones.

This proves that $\mathcal{P}_2$ also needs pseudo-polynomial memory in $\mathsf{Energy}_{\mathsf{LU}}$ games. Finally, we remark that this reasoning also holds for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ objective with threshold $t \coloneqq U$, as for the previous game. □

C.5 Average-Energy with Lower-Bounded Energy

We conclude with the conjunction of an AE objective with a lower bound (again equal to zero) constraint on the running energy, but no upper bound. This corresponds to an *hypothetical* unbounded energy storage. Hence, its applicability is limited, but it may prove interesting on the theoretical standpoint.

Problem C.3 ($\mathsf{AvgEnergy}_{\mathsf{L}}$)**.** *Given a game $G(T)$, an initial state s_{init} and a threshold $t \in \mathbb{Q}$, decide if $\mathcal{P}_1$ has a winning strategy $\sigma_1 \in \Sigma_1$ for objective $\mathit{Energy}_L(c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t)$.*

This problem proves to be challenging to solve: we provide partial answers in the following, with a proper algorithm for one-player games but only a correct but incomplete method for two-player games. As usual, we present our results for the supremum variant $\overline{AE}$.

Illustration. Consider the game in Fig. C.3. Recall that for $\mathsf{AvgEnergy}_{\mathsf{LU}}$ with $U = 3$, the optimal play is π_3, and it requires alternation between all three different simple cycles. Now consider $\mathsf{AvgEnergy}_{\mathsf{L}}$. One may think that relaxing the objective would allow for simpler winning strategies. This is not the case. Some new plays are now acceptable w.r.t. the energy constraint, such as $\pi_4 = (aabaaba)^\omega$, with $\overline{AE}(\pi_4) = 11/7$ and $\pi_5 = (aaababa)^\omega$, with $\overline{AE}(\pi_5) = 18/7$. Yet, the optimal play w.r.t. the AE (under the lower-bound energy constraint) is still π_3, hence still requires to use all the available cycles, in the appropriate order. This indicates that $\mathsf{AvgEnergy}_{\mathsf{L}}$ games also require complex solutions.

C.5.1 One-player Games

We assume that the unique player is $\mathcal{P}_1$. Indeed, the opposite case is easy as for $\mathcal{P}_2$, the objective is a disjunction and $\mathcal{P}_2$ can choose beforehand which sub-

objective he will transgress, and do so with a simple memoryless strategy (both AE and $\mathsf{Energy}_{\mathsf{L}}$ games admit memoryless optimal strategies as seen before). We show that one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problems lie in PSPACEby reduction to $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problems for a well-chosen upper bound $U \in \mathbb{N}$ and then application of Thm. C.13.

The reduction. Given a game $G = (S_1, S_2 = \emptyset, E, w)$ with largest weight $W \in \mathbb{N}$, an initial state s_{init}, and a threshold $t \in \mathbb{Q}$, we reduce the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem to an $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem with an upper bound $U \in \mathbb{N}$ defined as $U \coloneqq t + N^2 + N^3$, with $N = W \cdot (|S| + 2)$. Observe that the length of the binary encoding of U is polynomial in the size of the game, the encoding of the largest weight W and the encoding of the threshold t. The intuition is that if $\mathcal{P}_1$ can win a one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ game, he can win it without ever reaching energy levels higher than the chosen bound U, even if he is technically allowed to do so. Essentially, the interest of increasing the energy is making more cycles available (as they become safe to take w.r.t. the lower bound constraint), but increasing the energy further than necessary is not a good idea as it will negatively impact the average-energy. To prove this reduction, we start from an arbitrary winning path in the $\mathsf{AvgEnergy}_{\mathsf{L}}$ game, and build a witness path that is still winning for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective, but also keeps the energy below U at all times. Our construction exploits a result of Lafourcade et al. that bounds the value of the counter along a path in a one-counter automaton (stated in [93] and proved in [92, Lem. 42]). We slightly adapt it to our framework in the next lemma. The technique is identical, but the statement is more precise. In the following, we call an *expanded configuration* of the game G a couple (s, c) where $s \in S$ is a state and $c \in \mathbb{Z}$ a level of energy.

Lemma C.16. *Let $g \in \mathbb{Z}$. Let (s, c) and (s', c') be two expanded configurations of the game G such that there exists an expanded path $\rho_{\text{exp}} = (s_0, c_0) \dots (s_m, c_m)$ in G from (s, c) to (s', c') with $c_i \geq g$ for every $0 \leq i \leq m$. Then, there is a path $\rho'_{\text{exp}} = (s'_0, c'_0)(s'_1, c'_1) \dots (s'_n, c'_n)$ in G from (s, c) to (s', c') such that:*

- *for every $0 \leq i \leq n$, $g \leq c'_i \leq \max\{c, c', g\} + N^2 + N^3$, where $N = W \cdot (|S| + 2)$, with W the maximal absolute weight in G;*
- *there is an (injective) increasing mapping $\iota \colon \{1, \dots, n\} \to \{1, \dots, m\}$ such that for every $1 \leq i \leq n$, $s'_i = s_{\iota(i)}$ and $c'_i \leq c_{\iota(i)}$.*

Furthermore, for any two expanded paths ρ^1 and ρ^2, with $\mathit{last}(\rho^1) = (s, c)$ and $\mathit{first}(\rho^2) = (s', c')$, if $AE(\rho^1 \cdot \rho_{\text{exp}} \cdot \rho^2) \leq g$, then it also holds that

$AE(\rho^1 \cdot \rho'_{\text{exp}} \cdot \rho^2) \leq AE(\rho^1 \cdot \rho_{\text{exp}} \cdot \rho^2) \leq g$.

Proof. We write $\alpha = W \cdot (|S| + 1)$, $\beta = (\alpha + W) \cdot (\alpha + W - 1) - 1$ and $K = \max\{c, c', g\} + (\alpha + W)^2$. We apply inductively a transformation that removes similar ascending and descending segments of the path. The segments are selected such that their composition is neutral w.r.t. the energy.

Pick a subpath $\rho_{\text{exp}}[k, k+h] = (s_k, c_k) \dots (s_{k+h}, c_{k+h})$ of ρ_{exp}, if it exists, such that:

(*a*) $c_k \leq K$ and $c_{k+h} \leq K$;

(*b*) for every $0 < \ell < h$, $c_{k+\ell} > K$;

(*c*) there is $0 < \ell < h$ such that $c_{k+\ell} > K + W \cdot (|S| + 1) \cdot \beta$.

If such a subpath does not exist, then this means that the cost along ρ_{exp} is overall bounded by $K + W \cdot (|S| + 1) \cdot \beta$ (since condition (*a*) is not restrictive– $c, c' \leq K$), which then concludes the proof. Hence, assume such a subpath exists for the following steps.

Ascent part. Let $k \leq \ell_0 \leq \cdots \leq \ell_\beta \leq k + h$ be indices such that:

- $c_{\ell_i} > K + i \cdot W \cdot (|S| + 1)$;
- for every $k \leq \ell < \ell_i$, $c_\ell \leq K + i \cdot W \cdot (|S| + 1)$.

Fix $0 \leq i \leq \beta$. Then it holds that $c_{\ell_i} \leq K + i \cdot W \cdot (|S| + 1) + W$ and thus $c_{\ell_{i+1}} - c_{\ell_i} > K + (i+1) \cdot W \cdot (|S| + 1) - (K + i \cdot W \cdot (|S| + 1) + W) = W \cdot (|S| + 1) - W = W \cdot |S|$. Let J_i be a subset of $[\ell_i; \ell_{i+1}]$ defined by $\ell_i \in J_i$, and if $j \in J_i$, then let $j' \leq \ell_{i+1}$ be the smallest index larger than j (if it exists) such that $c_{j'} > c_j$. Obviously we have $c_j < c_{j'} \leq c_j + W$. Hence the cardinal of J_i is at least $1 + \frac{W \cdot |S|}{W} \geq |S| + 1$. Hence there is a state $\widetilde{s}^{(i)}$ and two indices $j_{i,1} < j_{i,2} \in J_i$ with $(s_{j_{i,1}}, c_{j_{i,1}}) = (\widetilde{s}^{(i)}, \alpha_1)$ and $(s_{j_{i,2}}, c_{j_{i,2}}) = (\widetilde{s}^{(i)}, \alpha_2)$ with $c_{\ell_i} \leq \alpha_1 < \alpha_2 \leq c_{\ell_{i+1}}$, hence using previous computed bounds, $0 < \alpha_2 - \alpha_1 \leq c_{\ell_{i+1}} - c_{\ell_i} < W \cdot (|S| + 2) = \alpha + W$. We write $\widetilde{d}^{(i)} = \alpha_2 - \alpha_1$. The segment between indices $j_{i,1}$ and $j_{i,2}$ is a candidate for being removed. Due to the value of β, there is $d \in \{\widetilde{d}^{(i)} \mid 0 \leq i \leq \beta\}$ that appears $(\alpha + W)$ times in that set.

Descent part. We do a similar reasoning for the "descent" part. There must exist indices $k \leq m_0 \leq \cdots \leq m_\beta \leq k + h$ such that:

- $c_{m_i} > K + (\beta - i) \cdot W \cdot (|S| + 1)$;

- for every $m_i < m \leq k + h$, $c_m \leq K + (\beta - i) \cdot W \cdot (|S| + 1)$.

Note that we obviously have $\ell_\beta < m_0$.

Then we apply the same combinatorics as for the ascent part. There is some value $0 < d' < \alpha + W$ which appears at least $\alpha + W$ times in potential cycles within the segment $\rho_{\text{exp}}[k, k + h]$.

Transformation. The algorithm then proceeds by removing d' segments that increase the cost by d within $\rho_{\text{exp}}[\ell_0, \ell_\beta]$ and d segments that decrease the cost by d' within $\rho_{\text{exp}}[m_0, m_\beta]$. This yields another path ρ'_{exp} and an obvious injection of ρ'_{exp} into ρ_{exp} which satisfies all the mentioned constraints. The sum of all energy levels along ρ'_{exp} is smaller than that along ρ_{exp}, and any energy level along ρ'_{exp} is obtained from that along ρ_{exp} by decreasing by at most $0 < d \cdot d' < (\alpha + W)^2$. By assumption on segment $\rho_{\text{exp}}[k, k + h]$ and bound K, we get that the cost along ρ'_{exp} is always larger than or equal to g, c and c'.

We iterate this transformation to get a uniform upper bound. We finally notice that the obtained upper bound $K + W \cdot (|S| + 1) \cdot \beta$ is bounded itself by $\max\{c, c', g\} + N^2 + N^3$, where $N = W \cdot (|S| + 2)$. This implies the expected result. □

We build upon this lemma to define an appropriate transformation leading to the witness path and derive a sufficiently large upper bound $U \in \mathbb{N}$ for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem.

Lemma C.17. *The* $\mathsf{AvgEnergy}_{\mathsf{L}}$ *problem over a one-player game* $G = (S_1, S_2 = \emptyset, E, w)$*, with an initial state* s_{init} *and a threshold* $t \in \mathbb{Q}$*, is reducible to an* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *problem over the same game* G*, for the same threshold* t *and upper bound* $U \coloneqq t + N^2 + N^3$*, with* $N = W \cdot (|S| + 2)$*.*

Proof. We prove that we can bound the energy along a witness of the one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem. Let σ be a winning strategy of $\mathcal{P}_1$ for objective $\mathit{Energy}_L(c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t)$ and $\pi = s_0 s_1 \dots s_n \dots$ be the corresponding outcome.

We build another strategy $\widetilde{\sigma}$ with corresponding play $\widetilde{\pi}$ such that for every n, $0 \leq c_{\mathsf{init}} + \mathrm{EL}(\widetilde{\pi}(n)) \leq c_{\mathsf{init}} + t + N^2 + N^3$, where $N = W \cdot (|S| + 2)$ (W is the maximal absolute weight in G), and such that $\overline{AE}(\widetilde{\pi}) \leq \overline{AE}(\pi)$. We actually build the play $\widetilde{\pi}$ directly, and infer strategy $\widetilde{\sigma}$.

From π, we build the expanded play $\pi_{\text{exp}} = (s_0, c_0)(s_1, c_1)\dots(s_n, c_n)\dots$ such that $c_i = \text{EL}(\pi(i))$ for every $i \geq 0$. Since π is a witness satisfying the objective $\mathit{Energy}_L(c_{\mathsf{init}}) \cap \mathit{AvgEnergy}(t)$, it holds that $c_i + c_{\mathsf{init}} \geq 0$ for every $i \geq 0$. We now show that some pair (s, c) is visited infinitely often along π_{exp}. Toward a contradiction, assume that it is not the case. Then since energy levels are bounded from below along π, this means that $\liminf_{n\to\infty} c_n = \underline{TP}(\pi) = +\infty$, and by Lem. C.2, that $\overline{AE}(\pi) = +\infty$ which contradicts the play being winning for the AE objective with threshold $t \in \mathbb{Q}$. Now select the smallest energy c and state s such that (s, c) is visited infinitely often along π_{exp}. Pick n_0 such that (1) $(s_{n_0}, c_{n_0}) = (s, c)$, (2) $\pi[\geq n_0] = s_{n_0}s_{n_0+1}\dots$ only visits states that are visited infinitely often along π, and (3) for every (s', c') along $\pi_{\text{exp}}[\geq n_0]$, it holds that $c' \geq c$.

We can then write π_{exp} as $\pi_{\text{exp}}[\leq n_0] \cdot \mathcal{C}_1 \cdot \mathcal{C}_2 \dots$ where each $\mathcal{C}_i$ ends at configuration (s, c) (hence $\mathcal{C}_i$ forms a cycle), and each configuration (s', c') along some $\mathcal{C}_i$ satisfies $c' \geq c$. We write γ_i for the projection of $\mathcal{C}_i$ on states (without energy level) — it forms a cycle as well. We obviously have

$$\overline{AE}(\pi) = \text{EL}(\pi(n_0)) + \overline{AE}(\pi[> n_0]) = c + \overline{AE}(\pi[> n_0])$$

by Lem. C.4, and since $\overline{AE}(\pi) \leq t$, there must be some cycle $\mathcal{C}_i$ such that $AE(\gamma_i) \leq t - c$. We write γ for such a γ_i, and we define $\varpi = \pi(n_0) \cdot \gamma^\omega$: it is a lasso-shaped play which also satisfies the objective $\mathit{Energy}_L(c_{\mathsf{init}}) \cap \mathit{AvgEnergy}(t)$.

We will now modify the play ϖ, so that the energy does not grow too much along it. We write ϖ_{exp} for the expanded version of ϖ: it is of the form

$$\varpi_{\text{exp}}[\leq n_0] \cdot \Big(\varpi_{\text{exp}}[n_0 + 1, n_0 + p]\Big)^\omega,$$

where $\varpi_{\text{exp}}[n_0{+}1, n_0{+}p]$ projects onto γ when the energy information is removed (note that the last configurations of $\varpi_{\text{exp}}[\leq n_0]$ and of $\varpi_{\text{exp}}[n_0 + 1, n_0 + p]$ are (s, c)). We will do two things: (i) first we will work on the cycle γ; and (ii) then we will work on the prefix $\varpi[\leq n_0]$, to build a witness with a fixed upper bound on the energy. For the rest of the proof, we assume that $\varpi_{\text{exp}} = (s_0, c_0)(s_1, c_1)\dots$ so that $(s_n, c_n) = (s, c)$ for every $n = n_0 + b \cdot p$ for some integer b.

First consider point (i). Let us notice that $c \leq t$, otherwise the average-energy along ϖ could not be at most t (remember that the cost along the expanded version of γ starting at (s, c) is always larger than or equal to c by construction). We pick the first maximal subpath $\varpi_{\text{exp}}[k, k + h]$ of ϖ_{exp} with $[k, k+h] \subseteq (n_0, n_0+p)$, such that $c_{k+\ell} > t$ for every $0 \leq \ell \leq h$. By maximality of $\varpi_{\text{exp}}[k, k + h]$, it is the case that $c_{k-1} \leq t$ and $c_{k+h+1} \leq t$. We infer that

$t < c_k \leq t + W$ and $t < c_{k+h} \leq t + W$, where W is the maximal absolute weight in the game G. We apply Lem. C.16 to the path $\varpi_{\text{exp}}[k, k+h]$ with $g = t$, and we get that we can build an expanded path $\varpi_{\text{exp}}^{(k)}$ which is shorter than $\varpi_{\text{exp}}[k, k+h]$ and such that:

- at all positions of $\varpi_{\text{exp}}^{(k)}$, the energy is in the interval $[t, t + N^2 + N^3]$, where $N = W \cdot (|S| + 2)$;
- there is an injective increasing mapping $\iota \colon [0, |\varpi_{\text{exp}}^{(k)}|] \to [k, k+h]$ such that for every index $1 \leq i \leq |\varpi_{\text{exp}}^{(k)}|$, the state of $\varpi_{\text{exp}}^{(k)}[= i]$ coincides with that of $\varpi_{\text{exp}}[= \iota(i)]$ and the energy at position i of $\varpi_{\text{exp}}^{(k)}$ is smaller than or equal to $c_{\iota(i)}$.

In particular, we have a new witness for the objective $\mathit{Energy}_L(c_{\mathsf{init}}) \cap \mathit{AvgEnergy}(t)$, which is the play $\varpi[< n_0] \cdot \Big(\varpi[n_0, k-1] \cdot \varpi^{(k)} \cdot \varpi[k+h+1, n_0 + |\gamma| - 1]\Big)^{\omega}$, where $\varpi^{(k)}$ is the projection of $\varpi_{\text{exp}}^{(k)}$ over the states of the game G. We iterate this transformation over all relevant segments of γ (this will happen only a finite number of times), and we end up with a new lasso-play $\varpi' = \varpi[\leq n_0] \cdot (\gamma')^{\omega}$ such that:

- ϖ' satisfies the objective $\mathit{Energy}_L(c_{\mathsf{init}}) \cap \mathit{AvgEnergy}(t)$;
- for every $1 \leq \ell \leq |\gamma'|$, $-c_{\mathsf{init}} \leq \mathrm{EL}(\varpi'(n_0 + \ell)) \leq t + N^2 + N^3$.

Now, consider point (ii). It remains to work on the prefix $\varpi[\leq n_0]$ (which is still a prefix of ϖ'). We apply Lem. C.16 to the prefix $\varpi[\leq n_0]$ with $g = 0$, and we get an appropriately bounded witness.

Summing up, our construction proves that if there exists a winning play for $\mathit{Energy}_L(c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t)$ in the one-player game G, then there exists one for $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t)$, with $U \coloneqq t + N^2 + N^3$. Since the converse implication is obvious (as the second objective is strictly stronger), this concludes the proof of the reduction to an $\mathsf{AvgEnergy}_{\mathsf{LU}}$ game. □

Complexity. Plugging this bound U in the PSPACEalgorithm for one-player $\mathsf{AvgEnergy}_{\mathsf{LU}}$ games (Thm. C.13) implies PSPACE-membership for one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games also. In terms of time complexity, we saw that this problem can thus be solved in pseudo-polynomial time. We prove that no truly-polynomial-time algorithm can be obtained unless PTIME = NP as the one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem is NP-hard. We show it by reduction from the subset-sum problem [65]: given a finite set of naturals $A = \{a_1, \ldots, a_n\}$ and a target natural v, decide if there exists a subset $B \subseteq A$ such that $\sum_{a_i \in B} a_i = v$. The reduction is sketched in Fig. C.6: a play corresponds to a choice of subset.

In order to keep a positive energy level, $\mathcal{P}_1$ has to pick a subset that achieves a sum *at least* equal to v, but in order to satisfy the AE threshold, this sum must be *at most* v: hence $\mathcal{P}_1$ must be able to pick a subset whose sum is *exactly* the target v.

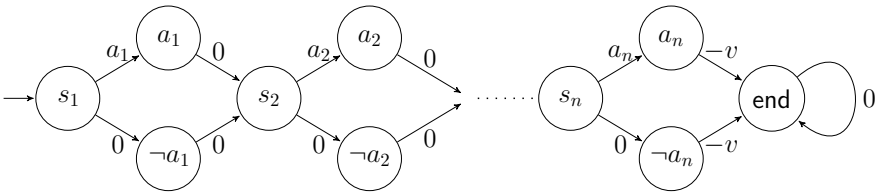


Figure C.6: Reduction from the subset-sum problem for target $v \in \mathbb{N}$ to a one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem for average-energy threshold $t \coloneqq v$.

Theorem C.18. *The* $\mathsf{AvgEnergy}_{\mathsf{L}}$ *problem is in* PSPACE *and at least* NP*-hard for one-player games.*

Proof. First, consider the claim of PSPACE-membership. Let $G = (S_1, S_2 = \emptyset, E, w)$ be a game with initial state s_{init}. Consider the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem for a given average-energy threshold $t \in \mathbb{Q}$. By Lem. C.17, this problem is reducible to the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem with upper bound $U \coloneqq t + N^2 + N^3$, with $N = W \cdot (|S| + 2)$. Hence, U is of order $\mathcal{O}(t + W^3 \cdot |S|^3)$, and its encoding is polynomial in the encoding of the original $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem (including thresholds and weights, not only in the number of states of the original game!). Following the complexity analysis presented in Thm. C.13, we thus conclude that the one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem is indeed in PSPACE. In terms of time, by using the MeanPayoff reduction and the pseudo-polynomial algorithm, we have an algorithm for the one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem that takes time of order

$$\mathcal{O}\left(\left((U+1)\cdot|S|+1\right)^3 \cdot \max\{U, \lceil t\rceil + 1\}\right) = \mathcal{O}\left(\left(t + W^3\cdot|S|^3\right)^4 \cdot |S|^3\right),$$

which is still pseudo-polynomial in the size of the original $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem (i.e., polynomial in the number of states and in the values of the largest absolute weight and of the average-energy threshold).

Second, we prove that the one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem is NP-hard. Consider the subset-sum problem for the set $A = \{a_1, \ldots, a_n\}$ such that for all $i \in \{1, \ldots, n\}$, $a_i \in \mathbb{N}$, and target $v \in \mathbb{N}$. Deciding if there exists a subset $B \subseteq A$ such that $\sum_{a_i \in B} a_i = v$ is well-known to be NP-complete [65]. We reduce this problem to an $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem over the game G depicted in Fig. C.6. Observe that this game has polynomially as many states as the size of A, and that its largest absolute weight is equal to the maximum between the largest element of A and the target v. It is clear that there is a bijection between

choices of subsets of A and plays in G. Let us fix threshold $t \coloneqq v$ for the average-energy. Recall that Lem. C.4 implies that the average-energy of any play is exactly its energy level at the first visit of end (because afterwards the zero self-loop is repeated forever). Hence, we have that

1. a play π in G is winning for $\mathit{Energy}_L(c_{\mathsf{init}} \coloneqq 0)$ if and only if the corresponding subset B is such that $\sum_{a_i \in B} a_i \geq v$;
2. a play π in G is winning for $\mathit{AvgEnergy}(t \coloneqq v)$ if and only if the corresponding subset B is such that $\sum_{a_i \in B} a_i \leq v$.

Therefore, $\mathcal{P}_1$ has a winning strategy for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective $\mathit{Energy}_L(c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t \coloneqq v)$ in G if and only if there exists a subset B for which the sum of elements is exactly equal to the target v.

This proves the reduction from the subset-sum problem and the NP-hardness result. Observe two things. First, the hardness proof relies on having set elements and a target value that are not polynomial in the size of the input set A. Indeed, the subset-sum problem is solvable with a pseudo-polynomial algorithm, hence in PTIME for polynomial values. Second, our reduction also holds for the $\underline{AE}$ variant of the average-energy. □

Memory requirements. Recall that for $\mathcal{P}_2$, the situation is simpler and memoryless strategies suffice. By the reduction to $\mathsf{AvgEnergy}_{\mathsf{LU}}$, we know that pseudo-polynomial memory suffices for $\mathcal{P}_1$. This bound is tight as witnessed by the family of games already presented in Fig. C.5a. To ensure the lower bound on energy, $\mathcal{P}_1$ has to play edge (s, s') at least U times before taking the (s, s) self-loop. But to minimize the average-energy, edge (s, s') should never be played more than necessary. The optimal strategy is the same as for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem: playing (s, s') exactly U times, then (s, s) once, then repeating, forever. As shown in Thm. C.15, this strategy requires pseudo-polynomial memory.

Theorem C.19. *Pseudo-polynomial-memory strategies are both sufficient and necessary to win for $\mathcal{P}_1$ in one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games. Memoryless strategies suffice for $\mathcal{P}_2$ in such games.*

C.5.2 Two-player Games

For the two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem, we only provide partial answers, as open questions remain. We first discuss decidability: we present an incremental algorithm that is correct but incomplete (Lem. C.20) and we draw the outline

of a potential approach to obtain completeness hence decidability. Then, we prove that the two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem is at least EXPTIME-hard (Lem. C.21). Finally, we show that in contrast to the one-player case, $\mathcal{P}_2$ also requires memory in two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games (Lem. C.22).

Decidability. Assume that there exists some $U \in \mathbb{N}$ such that $\mathcal{P}_1$ has a winning strategy for the $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem with upper bound U and average-energy threshold t. Then, this strategy is trivially winning for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem as well. This observation leads to an incremental algorithm that is correct (no false positives) but incomplete (it is not guaranteed to stop).

Lemma C.20. *There is an algorithm that takes as input an* $\mathsf{AvgEnergy}_{\mathsf{L}}$ *problem and iteratively solves corresponding* $\mathsf{AvgEnergy}_{\mathsf{LU}}$ *problems for incremental values of* $U \in \mathbb{N}$*. If a winning strategy is found for some* $U \in \mathbb{N}$*, then it is also winning for the original* $\mathsf{AvgEnergy}_{\mathsf{L}}$ *problem. If no strategy is found up to value* $U \in \mathbb{N}$*, then no strategy of* $\mathcal{P}_1$ *can simultaneously win the* $\mathsf{AvgEnergy}_{\mathsf{L}}$ *problem and prevent the energy from exceeding* U *at all times.*

While an incomplete algorithm clearly seems limiting from a theoretical standpoint, it is worth noting that in practice, such approaches are common and often necessary restrictions, even for problems where a complete algorithm is known to exist. For example, the existence of an initial energy level sufficient to win in multi-dimensional energy games can be decided [36] but practical implementations resort to an incremental scheme that is in practice incomplete because the theoretical bound granting completeness is too large to be tackled efficiently by software synthesis tools [14]. In our case, we have already seen that if such a bound exists for the two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem, it needs to be at least exponential in the encoding of problem (cf. one-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games). Hence it seems likely that a prohibitive bound would be necessary, rendering the algorithm of Lem. C.20 more appealing in practice.

Nevertheless, we *conjecture* that the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem is decidable for two-player games and that, similarly to the one-player case, an upper bound on the energy can be obtained. Unfortunately, this claim is much more challenging to prove for two-player games. Clearly, the approach of Lem. C.17 has to be generalized: while in one-player games we could pick a witness winning play and transform it, we now have to deal with *tree unfoldings* — describing sets of plays — because of the uncontrollable choices made by $\mathcal{P}_2$.

A potentially promising approach is to define a notion close to the *self-covering trees* used in [36] for energy games. Roughly, take any winning strategy of $\mathcal{P}_1$ in a two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ game. Without further assumption, this strategy

could be infinite-memory. It can be represented by its corresponding infinite tree unfolding where in nodes of $\mathcal{P}_1$, a unique child is given by the strategy, and in nodes of $\mathcal{P}_2$, all possible successors yield different branches. Every rooted branch of this tree is infinite and describes a winning play. Then, we would like to achieve the following steps.

1. Prove that all branches of this unfolding can be cut in such a way that the resulting finite tree describes a *finite-memory* strategy that is still winning for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective.
2. Reduce the height of this finite tree by compressing parts of the branches: deleting embedded zero cycles seems to be a good candidate for the transformation to apply.
3. Derive an *upper bound* on the height of the compressed tree and, consequently, on the maximal energy level reached along any play consistent with the corresponding strategy.
4. Use this upper bound to reduce the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem to an $\mathsf{AvgEnergy}_{\mathsf{LU}}$ problem.

Sadly, some challenges appear on the technical side when trying to implement this approach, mainly for items 1 and 3. Intuitively, the additional difficulty (when compared to the approach developed in [36] and similar works) arises from the fact that describing what is a good cycle pattern for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective is much more intricate than it is for a simple $\mathsf{Energy}_{\mathsf{L}}$ objective (in which case we simply look for zero cycles). This makes the precise definition of an appropriate transformation of branches, and the resulting tree height analysis, more tedious to achieve.

We also mention that the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem could be reduced, following a construction similar to the one given in Sect. C.4.1, to a mean-payoff threshold problem over an infinite arena, where states of the expanded graph are arranged respectively to their energy level, ranging from zero to infinity, and where weights would also take values inside $\mathbb{N} \cup \{\infty\}$ (as they reflect the possible energy levels). To the best of our knowledge, it is not known if mean-payoff games over such particular structures are decidable. If so, an algorithm would have to fully exploit the peculiar form of those arenas, as it is for example known that general models such as pushdown games are undecidable for the mean-payoff [37].

Finally, one could envision to fill the gap between one-player and two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games by using a general result similar to [69, Cor. 7]. Recall that we used it to derive memoryless determinacy in the two-player case from

memoryless determinacy of both one-player versions ($S_1 = \emptyset$ and $S_2 = \emptyset$). However, we here have that in one-player games, $\mathcal{P}_1$ requires pseudo-polynomial memory. Therefore, it is necessary to extend the result of Gimbert and Zielonka to finite-memory strategies: that is, to show that if we have a bound on memory valid in both one-player versions of a game, then this bound, or a derived one, is also valid in the two-player version. This is not known to be the case in general, and establishing it for a sufficiently general class of games seems challenging.

Complexity lower bound. We now prove that the two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem would require at least exponential time to solve. Our proof is by reduction from *countdown games*. A countdown game $\mathcal{C}$ is a weighted graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the finite set of states, and $\mathcal{E} \subseteq \mathcal{V} \times \mathbb{N} \setminus \{0\} \times \mathcal{V}$ is the edge relation. Configurations are of the form (v, c), $v \in \mathcal{V}$, $c \in \mathbb{N}$. The game starts in an initial configuration (v_{init}, c_0) and transitions from a configuration (s, c) are performed as follows. First, $\mathcal{P}_1$ chooses a duration d, $0 < d \leq c$ such that there exists $e = (v, d, v') \in \mathcal{E}$ for some $v' \in \mathcal{V}$. Second, $\mathcal{P}_2$ chooses a state $v' \in \mathcal{V}$ such that $e = (v, d, v') \in \mathcal{E}$. Then the game advances to $(v', c - d)$. Terminal configurations are reached whenever no legitimate move is available. If such a configuration is of the form $(v, 0)$, $\mathcal{P}_1$ wins the play, otherwise $\mathcal{P}_2$ wins. Deciding the winner given an initial configuration (v_{init}, c_0) is EXPTIME-complete [80].

Our reduction is depicted in Fig. C.7. The EL is initialized to c_0, then it is decreasing along any play. Consider the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective for AE threshold $t \coloneqq 0$. To ensure that the energy always stays non-negative, $\mathcal{P}_1$ has to switch to stop while the EL is no less than zero. In addition, to ensure an AE no more than $t = 0$, $\mathcal{P}_1$ has to obtain an EL at most equal to zero before switching to stop (as the AE will be equal to this EL thanks to Lem. C.4 and the zero self-loop on stop). Hence, $\mathcal{P}_1$ wins the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective only if he can ensure a total sum of chosen durations that is *exactly* equal to c_0, i.e., if he can reach a winning terminal configuration for the countdown game. The converse also holds.

Lemma C.21. *The* $\mathsf{AvgEnergy}_{\mathsf{L}}$ *problem is EXPTIME-hard for two-player games.*

Proof. Given a countdown game $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ and an initial configuration (v_{init}, c_0), we build a game $G = (S_1, S_2, E, w)$ with initial state s_{init} such that $\mathcal{P}_1$ has a winning strategy in G for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective for threshold $t \coloneqq 0$ if and only if he has a winning strategy in $\mathcal{C}$ to reach a terminal configuration

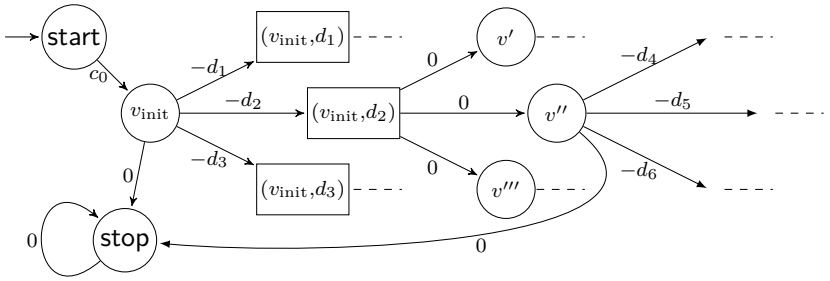


Figure C.7: Reduction from a countdown game $\mathcal{C} = (\mathcal{V}, \mathcal{E})$ with initial configuration (v_{init}, c_0) to a two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem for average-energy threshold $t \coloneqq 0$.

with counter value zero. The construction is depicted in Fig. C.7. Formally, the game G is built as follows.

- $S_1 = \mathcal{V} \cup \{\mathsf{start}, \mathsf{stop}\}$.
- $S_2 = \{(v, d) \in \mathcal{V} \times \mathbb{N} \setminus \{0\} \mid \exists\, v' \in \mathcal{V},\ (v, d, v') \in \mathcal{E}\}$.
- $s_{\text{init}} = \mathsf{start}$.
- For each $(v, d, v') \in \mathcal{E}$, we have that $(v, (v, d)) \in E$ with $w(v, (v, d)) = -d$ and $((v, d), v') \in E$ with $w((v, d), v') = 0$.
- Additionally, $(\mathsf{start}, v_{\text{init}}) \in E$ with $w(\mathsf{start}, v_{\text{init}}) = c_0$, $(\mathsf{stop}, \mathsf{stop}) \in E$ with $w(\mathsf{stop}, \mathsf{stop}) = 0$ and for all $v \in \mathcal{V}$, $(v, \mathsf{stop}) \in E$ with $w(v, \mathsf{stop}) = 0$.

First, consider the left-to-right direction of the claim. Assume $\mathcal{P}_1$ has a winning strategy for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective in G. As noted before, such a strategy necessarily reaches the energy level zero then switches to stop directly. Hence, applying this strategy in the countdown game ensures that the sum of durations will be exactly equal to c_0 (recall that we start our $\mathsf{AvgEnergy}_{\mathsf{L}}$ game by initializing the energy to c_0 then decrease it at every step by the duration chosen by $\mathcal{P}_1$). Thus, this strategy is winning in the countdown game $\mathcal{C}$.

Second, consider the right-to-left direction. Assume that $\mathcal{P}_1$ has a winning strategy in the countdown game $\mathcal{C}$. Playing this strategy in G ensures to reach a state $v \in S_1$ with energy level exactly equal to zero. Thus a winning strategy for the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective is to play the countdown strategy up to this point then to immediately take the edge (v, stop). Indeed, any consistent outcome will satisfy the lower bound on energy (as the energy will never go below zero), and it will have an average-energy equal to $t = 0$ (because the energy level when reaching stop will be zero).

This shows both directions of the claim and concludes our proof. Observe

that this reduction is also true if we consider the $\underline{AE}$ variant of the average-energy. □

Memory requirements. We close our study of two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games by discussing the memory needs. First note that we cannot provide upper bounds: if we had such bounds, we could derive a bound on the energy along any consistent play and reduce the $\mathsf{AvgEnergy}_{\mathsf{L}}$ problem to an $\mathsf{AvgEnergy}_{\mathsf{LU}}$ one as discussed before, hence proving its decidability. Second, we already know by Thm. C.19 that pseudo-polynomial memory is necessary for $\mathcal{P}_1$. Finally, we present a simple game (Fig. C.8) where $\mathcal{P}_2$ needs to use memory in order to prevent $\mathcal{P}_1$ from winning.

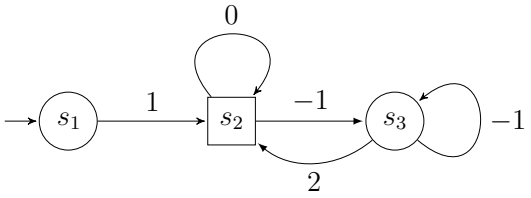


Figure C.8: Simple two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ game witnessing the need for memory even for $\mathcal{P}_2$.

Lemma C.22. *Pseudo-polynomial-memory strategies are necessary to win for $\mathcal{P}_1$ in two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games. Memory is also required for $\mathcal{P}_2$ in such games.*

Proof. We only have to prove that $\mathcal{P}_2$ needs memory in the game of Fig. C.8. Consider the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective for the average-energy threshold $t \coloneqq 1$ on this game. Assume that $\mathcal{P}_2$ is restricted to *memoryless* strategies. Then, there are only two possible strategies for $\mathcal{P}_2$. If $\mathcal{P}_2$ always takes the self-loop (s_2, s_2), then the only consistent play is $s_1(s_2)^\omega$: it has AE equal to 1, and satisfies the lower bound constraint on energy, thus $\mathcal{P}_1$ wins. If $\mathcal{P}_2$ always takes (s_2, s_3), then $\mathcal{P}_1$ can win by producing the following play: $s_1 s_2 (s_3 s_2 s_3)^\omega$. It also has AE equal to 1, and satisfies the energy constraint. Hence $\mathcal{P}_2$ cannot win this game with a memoryless strategy. Nonetheless, he has a winning strategy that uses memory. Let this strategy be the one that plays (s_2, s_3) once then chooses the self-loop (s_2, s_2) forever. When this strategy is used by $\mathcal{P}_2$, $\mathcal{P}_1$ has to pick (s_3, s_2) in the first visit of s_3 otherwise he loses because the energy goes below zero. But if $\mathcal{P}_1$ picks this edge, the unique outcome becomes $s_1 s_2 s_3 (s_2)^\omega$, whose average-energy is $2 > t$, hence also losing for $\mathcal{P}_1$. Thus, the defined strategy is winning for $\mathcal{P}_2$. □

C.6 Conclusion

We presented a thorough study of the *average-energy* payoff. We showed that average-energy games belong to the same intriguing complexity class as mean-payoff, total-payoff and energy games and that they are similarly memoryless determined. We then solved average-energy games with lower- and upper-bounded energy: such a conjunction is motivated by previous case studies in the literature [29]. Lastly, we provided preliminary results for the case of average-energy with a lower bound but no upper bound on the energy. Following the publication of [19], Larsen et al. adressed a different problem in [94]: they proved that deciding if there exists a threshold $t \in \mathbb{Q}$ such that $\mathcal{P}_1$ can win a two-player game for objective $\mathit{Energy}_L(c_{\mathsf{init}} \coloneqq 0) \cap \mathit{AvgEnergy}(t)$ can be done in doubly-exponential time. This is indeed equivalent to deciding if there exists an upper-bound $U \in \mathbb{N}$ such that $\mathcal{P}_1$ can win for the objective $\mathit{Energy}_{LU}(U, c_{\mathsf{init}} \coloneqq 0)$, which is known to be in $\mathsf{2EXPTIME}$ [78]. Unfortunately, this approach does not help in solving Problem C.3, where the threshold $t \in \mathbb{Q}$ for the average-energy is part of the input: solving two-player $\mathsf{AvgEnergy}_{\mathsf{L}}$ games is still an open question.

We believe that the average-energy objective and its variations model relevant aspects of systems in practical applications as hinted by the aforementioned case study. Hence, we would like to extend our knowledge of this objective to more general models such as stochastic games, or games with multi-dimensional weights. Of course, the open questions regarding the $\mathsf{AvgEnergy}_{\mathsf{L}}$ objective are intriguing. Finally, we would like to implement our techniques in synthesis tools and assess their applicability through proper case studies.

Paper D

Limit Your Consumption! Finding Bounds in Average-energy Game

Simon Laursen Kim G. Larsen
Aalborg University, Department of Computer Science, Denmark
Martin Zimmermann
Reactive Systems Group, Saarland University, Germany

Abstract Energy games are infinite two-player games played in weighted arenas with quantitative objectives that restrict the consumption of a resource modeled by the weights, e.g., a battery that is charged and drained. Typically, upper and/or lower bounds on the battery capacity are part of the problem description. Here, we consider the problem of determining upper bounds on the average accumulated energy or on the capacity while satisfying a given lower bound, i.e., we do not determine whether a given bound is sufficient to meet the specification, but if there exists a sufficient bound to meet it. We show that the problem of determining the existence of a sufficient bound on the long-run average accumulated energy can be solved in doubly-exponential time. Then, we consider recharge games: here, all weights are negative, but there are recharge edges that recharge the energy to some fixed capacity. We show that bounding the long-run average energy in such games is complete for exponential time. Then, we consider the existential version of the problem, which turns out to be solvable in polynomial time: here, we ask whether there is a recharge capacity that allows the system player to win the game.

Publication History The paper has been accepted for the 24th International Workshop on Quantitative Aspects of Programming Languages and Systems, EPTCS, by the Open Publishing Association 2016. The version included in this thises is a full version of the paper with all proves and a modified layout.

The layout has been revised.

D.1 Introduction

Quantitative games provide a natural framework for synthesizing controllers with resource restrictions and for performance requirements for reactive systems with an uncontrollable environment. In a traditional two-player graph game of infinite duration (see [70]), two players, Player 0 (who represents the system to be synthesized) and Player 1 (representing the antagonistic environment), construct an infinite path by moving a pebble through a graph, which describes the interaction between the system and its environment. The objective, which encodes the controller's specification, determines the winner of such a play. Quantitative games extend classical ones by having weights on edges for modeling costs, consumption or rewards, and a quantitative objective to encode the specification in terms of the weights.

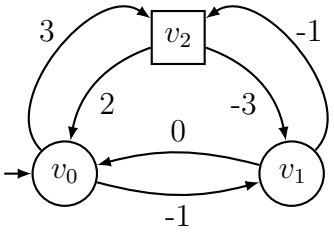


Figure D.1: Example of a game arena.

Consider the game depicted on Figure D.1: we interpret negative weights as energy consumption and correspondingly positive weights as recharges. Then, Player 0 (who moves the pebble at the circled vertices) can always maintain an energy level (the sum of the weights seen along a play prefix starting with energy 0) between zero and five using the following strategy: when at vertex v_0 with non-zero energy level go to vertex v_1, otherwise go to vertex v_2 in order to satisfy the lower bound. At vertex v_1 she moves to v_0 if the energy level is zero, otherwise to v_2. It is straightforward to verify that the strategy has the desired property when starting at the initial vertex v_0 with initial energy 0. However, this strategy requires memory to implement, as its choices depend on the current energy level.

Quantitative games [12, 31, 111] and objectives such as mean-payoff [26, 121, 126], energy [20, 39, 78], and their combination [43] have attracted considerable attention recently. The focus has been on establishing the computational complexity of deciding whether Player 0 wins the game and on memory requirements. In mean-payoff games, Player 0's goal is to optimize the long-run average gain per edge taken, whereas in energy games the goal is to keep

the accumulated energy within given bounds. Recently, the average-energy objective was introduced [21] to capture the specification in an industrial case study [29]. In this study, the authors synthesize a controller to operate an oil pump using timed games and Uppaal TiGA. The controller has to keep the amount of oil in an accumulator within given bounds while minimizing the average amount of oil in the accumulator in the long run. A discrete version of this problem is exactly an average-energy game, where the goal for Player 0 is to optimize the long-run average accumulated energy during a play while keeping the accumulated energy within given bounds.

Recall the introductory example above. The strategy for Player 0 described there realizes the long-run average 4: the consistent play $v_0(v_2v_0v_1)^\omega$ with energy levels $0, (3, 5, 4)^\omega$ has average 4, obtained by dividing the sum of the levels in the period by the length of the period. Every other consistent play has a smaller or equal average.

The computational complexity of these quantitative objectives are typically studied with respect to given bounds on the energy level or given thresholds on the mean-payoff or on the average accumulated energy. In this work, we consider the variants where the bounds and thresholds are existentially quantified instead of given as part of the input, i.e., we ask if there exist bounds and thresholds such that Player 0 has a winning strategy. This question is natural for models with bounds and thresholds as it desirable to know if a given model is realizable for some bounds. In a second step, one would then determine the minimal bounds for which Player 0 is able to win.

In particular, we study existential questions on two different game models, average-energy games and average-bounded recharge games. Average-energy games are defined as in [21] with both positive and negative weights on edges whereas in average-bounded recharge games all weights are negative, but there are designated recharge-edges that recharge the energy to some fixed capacity.

Our contribution. For average-energy games, we show that the problem of deciding whether there exists a threshold to which Player 0 can bound the long-run average accumulated energy while keeping the accumulated energy non-negative can be solved in doubly-exponential time. To this end, we show that the problem is equivalent to determining whether the maximal energy level can be uniformly bounded by a strategy. The latter problem is known to be in 2EXPTIME [78]. The challenging part is to construct a strategy that uniformly bounds the energy from the strategy that only bounds the long-run average accumulated energy, but might reach arbitrarily high energy levels. But whenever the energy level increases above the given threshold, it has to

drop below it at some later point. Thus, we can always play like in a situation where the peak between these two threshold crossings is as small as possible. This yields a new strategy that bounds the energy level. Our result is one step towards solving the open problem of solving lower-bounded average-energy games with a given threshold [21].

For average-bounded recharge games, we show that given a bound on the long-run average energy, deciding the winner is EXPTIME-complete. For the existential versions of the problem, we show that it remains EXPTIME-hard when the recharge capacity is quantified and the average threshold is given. The problem becomes solvable in polynomial time when only the recharge capacity is considered: here, we ask whether there is a recharge capacity such that Player 0 wins the game with respect to this capacity.

Finally, we study tradeoffs between the different bounds and the memory requirements of winning strategies, and show that increasing the upper bound on the maximal energy level allows to improve the average energy level and memory can be traded for smaller upper bounds and vice versa.

Related Work. The average energy objective was first introduced in [120] under the name total-reward but has until recently not undergone a systematic study. Independently, it was studied (under the name total-payoff) for Markov decision processes and stochastic games [17], and [21] presented a comprehensive investigation into the problem in the deterministic case. The latter also considered extensions where the average-energy objective is combined with bounds on the energy, which is the model we consider here.

Several other games with combined objectives have been introduced such as mean-payoff parity [42], energy-parity [39], multi-dimensional energy [59], multi-dimensional mean-payoff [121] and the combination of multi-dimensional energy, mean-payoff and parity [43]. In [24], consumption games are studied where edges only have negative weights, and some distinguished edges recharge the energy to a level determined by Player 0. This model is related to recharge games, but in recharge games the recharge capacity is given and we consider average-bounded objectives. Existential questions in games have been studied before in the form of determining the emptiness of a set of bounds that allow Player 0 to win a quantitative game, e.g., for multi-dimensional energy games with upper bounds [78] and for games with objectives in parameterized generalizations of LTL [1, 60, 91, 125].

D.2 Definitions

An *arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$ consists of a finite directed graph (V, E) without terminal vertices, a partition $V = V_0 \uplus V_1$ of the vertices, and an initial vertex $v_I \in V$. Vertices in V_0 are under Player 0's control and are drawn as circles, whereas vertices in V_1 are under Player 1's control and drawn as rectangles. A play in A is an infinite path $\pi = v_0 v_1 v_2 \cdots$ with $v_0 = v_I$. A *game* $\mathcal{G} = (\mathsf{A}, \mathrm{Win})$ consists of an arena A, and a set $\mathrm{Win} \subseteq V^\omega$ of winning plays for Player 0, the *objective* of $\mathcal{G}$. The objectives we consider are induced by weight functions, assigning integer weights to edges, which are encoded in binary. We say an algorithm runs in *pseudo-polynomial time*, if it runs in polynomial time in the number of vertices and in the largest absolute weight. An algorithm runs in polynomial time, if it runs in polynomial time in the number of vertices and in the size of the encoding of the largest absolute weight.

A *strategy* for Player $i \in \{0, 1\}$ is a mapping $\sigma_i \colon V^* V_i \to V$ such that $(v, \sigma_i(wv)) \in E$ for all $wv \in V^* V_i$. A play $v_0 v_1 v_2 \cdots$ is *consistent* with a strategy σ_i for Player i if $v_{n+1} = \sigma_i(v_0 v_1 \cdots v_n)$ for every n with $v_n \in V_i$. A strategy σ_0 for Player 0 is winning for the game $\mathcal{G} = (\mathsf{A}, \mathrm{Win})$ if every play that is consistent with σ_0 is in Win. We say that Player 0 wins $\mathcal{G}$ if she has a winning strategy for $\mathcal{G}$. We define $\mathit{Prefs}(\sigma)$ to denote the set of finite play prefixes that are consistent with σ. We denote the last vertex of a non-empty word w by $\mathit{last}(w)$.

A *memory structure* $\mathcal{M} = (M, m_I, \mathrm{Upd})$ for an arena (V, V_0, V_1, E, v_I) consists of a finite set M of memory states, an initial memory state $m_I \in M$, and an update function $\mathrm{Upd} \colon M \times E \to M$. The update function can be extended to $\mathrm{Upd}^+ \colon V^+ \to M$ in the usual way: $\mathrm{Upd}^+(v_0) = m_I$ and $\mathrm{Upd}^+(v_0 \cdots v_n v_{n+1}) = \mathrm{Upd}(\mathrm{Upd}^+(v_0 \cdots v_n), (v_n, v_{n+1}))$. A next-move function (for Player i) $\mathrm{Nxt} \colon V_i \times M \to V$ has to satisfy $(v, \mathrm{Nxt}(v, m)) \in E$ for all $v \in V_i$ and all $m \in M$. It induces a strategy σ for Player i via $\sigma(v_0 \cdots v_n) = \mathrm{Nxt}(v_n, \mathrm{Upd}^+(v_0 \cdots v_n))$. A strategy is called *finite-state* (*positional*) if it can be implemented by a memory structure (with a single state). Intuitively, the next move of a positional strategy only depends on the last vertex of the play prefix. An arena $\mathsf{A} = (V, V_0, V_1, E, v_I)$ and a memory structure $\mathcal{M} = (M, m_I, \mathrm{Upd})$ for A induce the expanded arena $\mathsf{A} \times \mathcal{M} = (V \times M, V_0 \times M, V_1 \times M, E', (v_I, m_I))$ where $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $\mathrm{Upd}(m, (v, v')) = m'$. Each play $v_0 v_1 v_2 \cdots$ in A has a unique extended play $(v_0, m_0)(v_1, m_1)(v_2, m_2) \cdots$ in $\mathsf{A} \times \mathcal{M}$ defined by $m_0 = m_I$ and $m_{n+1} = \mathrm{Upd}(m_n, (v_n, v_{n+1}))$, i.e., $m_n = \mathrm{Upd}^+(v_0 \cdots v_n)$. A game $\mathcal{G} = (\mathsf{A}, \mathrm{Win})$ is *reducible* to $\mathcal{G}' = (\mathsf{A}', \mathrm{Win}')$ via $\mathcal{M}$, written $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, if $\mathsf{A}' = \mathsf{A} \times \mathcal{M}$ and every play π in $\mathcal{G}$ is won by the player

who wins the extended play π' in $\mathcal{G}'$, i.e., $\pi \in \mathrm{Win}$ if, and only if, $\pi' \in \mathrm{Win}'$.

Lemma D.1. *If $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ and Player i has a positional winning strategy for $\mathcal{G}'$, then she has a finite-state winning strategy for $\mathcal{G}$ which is implemented by $\mathcal{M}$.*

D.3 Finding Bounds in Average-energy Games

In this section, we study average-energy games with existentially quantified bounds on the average accumulated energy: our main theorem shows that these games are solvable in doubly-exponential time.

A weight function for an arena (V, V_0, V_1, E, v_I) is a function $W \colon E \to \mathbb{Z}$ mapping every edge to an integer weight. The energy level of a play prefix is the accumulated weight of its edges, i.e., $\mathrm{EL}(v_0 \cdots v_n) = \sum_{i=0}^{n-1} W(v_i, v_{i+1})$. We consider several objectives obtained by specifying upper and lower bounds on the energy level and on the long-run average accumulated energy.

- The lower-bounded energy objective requires Player 0 to keep the energy level non-negative:

$$\mathsf{Energy}_{\mathsf{L}}(W) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \forall n.\, 0 \leq \mathrm{EL}(v_0 \cdots v_n)\}$$

- The lower- and upper-bounded energy objective requires Player 0 to keep the energy level always between 0 and some given upper bound *cap*, the so-called capacity:

$$\mathsf{Energy}_{\mathsf{LU}}(W, \mathit{cap}) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \forall n.\, 0 \leq \mathrm{EL}(v_0 \cdots v_n) \leq \mathit{cap}\}$$

- The average-energy objective requires Player 0 to keep the long-run average of the accumulated energy below a given threshold t:

$$AE(W, t) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \limsup_{n \to \infty} \frac{1}{n} \sum\nolimits_{i=0}^{n-1} \mathrm{EL}(v_0 \cdots v_i) \leq t\}$$

- Also, we consider conjunctions of objectives, i.e., the lower-bounded average-energy objective

$$\mathsf{AvgEnergy}_{\mathsf{L}}(W, t) = \mathsf{Energy}_{\mathsf{L}}(W) \cap AE(W, t)$$

and the lower- and upper-bounded average-energy objective

$$\mathsf{AvgEnergy}_{\mathsf{LU}}(W, \mathit{cap}, t) = \mathsf{Energy}_{\mathsf{LU}}(W, \mathit{cap}) \cap AE(t).$$

Note that we always assume the initial energy level to be zero. This is not a restriction, as one can always add a fresh initial vertex with an edge to the old initial vertex that is labeled by the desired initial energy level. Similarly, one can reduce arbitrary non-zero lower bounds to the case of the lower bound being zero, which is the one we consider here.

Decidability of determining the winner of a game with lower-bounded average-energy objective with a given threshold t is an open problem [21]. To take a step towards solving this problem, we consider the existential variant of the problem, i.e., we ask whether there exists some threshold t such that Player 0 wins the game with objective $\mathsf{AvgEnergy}_{\mathsf{L}}(W, t)$:

Problem D.1. *Existence of a threshold in a lower-bounded average-energy game.*
***Input**: Arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$ *and* $W \colon E \to \mathbb{Z}$
***Question**: Exists a threshold* $t \in \mathbb{N}$ *s.t. Player* 0 *wins* $(\mathsf{A}, \mathsf{AvgEnergy}_{\mathsf{L}}(W, t))$*?*

We show that this problem is reducible to asking for the existence of an upper bound on the capacity *cap*. Note that such an upper bound also bounds the average accumulated energy. However, the converse is non-trivial as the average can be bounded while the energy level is unbounded. Formally, we consider the following problem:

Problem D.2. *Existence of an upper bound in a lower- and upper-bounded energy game.*
***Input**: Arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$ *and* $W \colon E \to \mathbb{Z}$
***Question**: Exists a capacity* $cap \in \mathbb{N}$ *s.t. Player* 0 *wins* $(\mathsf{A}, \mathsf{Energy}_{\mathsf{LU}}(W, cap))$*?*

The main theorem of this section shows that the existence of a threshold in a lower-bounded average-energy game can be checked in doubly-exponential time. Our choice of encoding the weights influences the complexity of the problem: if the weights are encoded in unary, then the complexity drops to EXPTIME. Furthermore, the problem is trivially at least as hard as solving mean-payoff games.

Theorem D.2. *The threshold problem for lower-bounded average-energy games is in 2EXPTIME.*

To prove this theorem, it suffices to show that Problem D.1 and Problem D.2 are equivalent, as the latter problem was shown to be in 2EXPTIME [78].

Lemma D.3. *Let* A *be an arena and let* W *be a weight function for* A*. Player* 0 *wins* $(\mathsf{A}, \mathsf{AvgEnergy}_{\mathsf{L}}(W, t))$ *for some* $t \in \mathbb{N}$ *if, and only if, Player* 0

wins $(\mathsf{A}, \mathsf{Energy}_{\mathsf{LU}}(W, cap))$ *for some* $cap \in \mathbb{N}$.

Proof. It is clear that a winning strategy σ for $(\mathsf{A}, \mathsf{Energy}_{\mathsf{LU}}(W, cap))$ for some $cap \in \mathbb{N}$ is a winning strategy for $(\mathsf{A}, \mathsf{AvgEnergy}_{\mathsf{L}}(W, cap))$, as if the energy level is always below some *cap*, then the average energy is also bounded by *cap*.

For the other direction, assume that σ is a winning strategy for Player 0 in $(\mathsf{A}, \mathsf{AvgEnergy}_{\mathsf{L}}(W, t))$ for some $t \in \mathbb{N}$. Now, we want to construct a strategy σ' that is winning for Player 0 in $(\mathsf{A}, \mathsf{Energy}_{\mathsf{LU}}(W, cap))$ for some $cap \in \mathbb{N}$. Note that σ might bound the average to some value while the energy level might be unbounded. But whenever the energy level increases above t, it has to drop below t at some point. We use this property to construct a strategy σ' that bounds the energy level.

First, we need to introduce some notation. Fix a play prefix $w \in \mathit{Prefs}(\sigma)$ with $\mathrm{EL}(w) > t$ and define

$$\mathrm{Peak}(w) = \sup\{\mathrm{EL}(wx) \mid wx \in \mathit{Prefs}(\sigma) \text{ and } \mathrm{EL}(wx') > t \text{ for all } x' \sqsubseteq x\},$$

i.e., $\mathrm{Peak}(w)$ is the supremum of the energy levels of prolongations of w that are consistent with σ and have not yet had an energy level below t. Applying König's Lemma [85] and the fact that σ is a winning strategy implies that the peak is always bounded.

Remark D.4. *We have* $\mathrm{Peak}(w) \in \mathbb{N}$ *for every* $w \in \mathit{Prefs}(\sigma)$.

For an energy level $c \in \mathbb{N}$ and a vertex $v \in V$ we define the set of possible ways to end up in vertex v with the energy level c playing consistently with σ as

$$\mathrm{Real}(v, c) = \{w \in \mathit{Prefs}(\sigma) \mid \mathit{last}(w) = v \text{ and } \mathrm{EL}(w) = c\}.$$

For every combination (v, c) with $c > t$, we pick a representative from $\mathrm{Real}(v, c)$ that minimizes the peak height among all such realizations, i.e., we define $\mathrm{Rep}(v, c)$ to be an element w from $\mathrm{Real}(v, c)$ with minimal peak-value $\mathrm{Peak}(w)$ among the play prefixes in $\mathrm{Real}(v, c)$. Note that $\mathrm{Rep}(v, c)$ might be undefined, i.e., if there is no play prefix ending in v with energy level c.

Intuitively, we construct a new strategy that mimics the behavior of σ until the energy level increases above t. At this point, the history is replaced by the representative for the last vertex and the current energy level. Then, our new strategy mimics the behavior of σ with this history until the threshold t is again crossed from below. Then, the next representative is picked. This strategy satisfies an upper bound, as only a finite number of representatives, each with

a bounded peak-value, are considered when mimicking σ. To formalize this, we recursively define $h\colon V^+ \to \mathit{Prefs}(\sigma)$ via $h(v_I) = v_I$ and

$$h(wv) = \begin{cases} \mathrm{Rep}(v, \mathrm{EL}(h(w)v)) & \text{if } \mathrm{EL}(h(w)) \leq t \text{ and } \mathrm{EL}(h(w)v) > t \\ h(w)v & \text{otherwise} \end{cases}$$

for a play prefix $wv \in V^+$ ending in a vertex v, i.e., $h(w)$ is the play prefix that simulates w. Now, we define the new strategy σ' via $\sigma'(w) = \sigma(h(w))$. The following remark implies that this is well-defined, although Rep and therefore h and σ might be undefined for certain inputs.

Remark D.5. *Let w be consistent with σ'. Then, $h(w)$ is defined and consistent with σ, $\mathit{last}(w) = \mathit{last}(h(w))$, and $\mathrm{EL}(w) = \mathrm{EL}(h(w))$.*

Applying the remark inductively we conclude that $h(w)$ is defined for every play prefix w that is consistent with σ'. This implies that $\sigma'(w)$ is well-defined for every such w that ends in a vertex from V_0. Furthermore, this also implies that σ' still satisfies the lower bound on the energy level.

Thus, it remains to prove that an upper bound exists. Let $\pi = v_0v_1v_2\cdots$ be consistent with σ' and let n be such that $\mathrm{EL}(v_0\cdots v_n) \leq t$ and $\mathrm{EL}(v_0\cdots v_nv_{n+1}) > t$. If there is no such n, then σ bounds the energy level by t and we are done. Furthermore, define n' to be minimal with $n' > n+1$ and $\mathrm{EL}(v_0\cdots v_{n'}) \leq t$ and $\mathrm{EL}(v_0\cdots v_{n'}v_{n'+1}) > t$ (if no such n' exists the reasoning is analogous). As the energy level between the positions $n+1$ and n' never crosses the threshold t from below, we are always in the second case of the definition of h. Thus, after the play prefix $v_0\cdots v_{n+1}$, the strategy σ' mimics the behavior of σ after the prefix $h(v_0\cdots v_{n+1}) = \mathrm{Rep}(v_{n+1}, \mathrm{EL}(v_0\cdots v_{n+1}))$. Therefore, the energy level between these two positions is bounded by $\mathrm{Peak}(\mathrm{Rep}(v_{n+1}, \mathrm{EL}(v_0\cdots v_{n+1})))$. As we only take those representatives into account that have an energy level between $t+1$ and $t+W$, where W is the largest positive weight in the image of W, the energy level of the play is bounded by the maximal peak of one of these representatives. Finally, this bound is uniform for all plays that are consistent with σ'. Thus, σ' is winning in the game $(\mathsf{A}, \mathsf{AvgEnergy}_{\mathsf{L}}(W, \mathit{cap}))$ for some cap. □

Note that we do not obtain any upper bounds on the energy level or on the long-run average energy realized by σ', as they depend on properties of σ. One can even construct examples that show these values to be arbitrarily large by starting with a *bad* winning strategy σ for the energy game.

D.4 Finding Bounds in Average-bounded Recharge Games

In this section, we study a variation of energy games called recharge games (the name is inspired by recharge automata, first introduced in [55]). In such games, there are designated recharge edges that recharge the energy to some given capacity. All other edges have non-positive cost, i.e., they only decrease the energy level or leave it unchanged. This is reminiscent of so-called consumption games [24], where Player 0 picks the new energy level while traversing a recharge edge. There, one is interested in which initial energy levels allow Player 0 to win and to compute upper bounds on the recharge levels picked by Player 0.

In this section, we go beyond just bounding the energy level by also considering bounds on the average accumulated energy, as we have done for average-energy games. However, the resulting games are intractable, as soon as the threshold on the average is part of the input. These results are presented in Subsection D.4.1. To overcome the high complexity, in Subsection D.4.2 we consider the problem where the recharge capacity is existentially quantified: this problem is solvable in polynomial time by a reduction to three-color parity games.

Here, we consider weight functions with only non-positive weights and a special recharge action R, i.e., $W\colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$. The recharge action R returns the energy level to some given upper bound capacity *cap*. The recharge energy level is the energy left since the last recharge action, which is defined as $\mathrm{EL}_{cap}(v_0 \cdots v_n) = cap + \mathrm{EL}(x)$, where x is the longest suffix of $v_0 \cdots v_n$ without an R-edge, i.e., $W(v_j, v_{j+1}) \neq \mathrm{R}$ for all (v_j, v_{j+1}) in x, which implies that a play starts with energy level *cap*. We define the objective of a recharge game as

$$\mathsf{Recharge}(W, cap) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \forall n.\, \mathrm{EL}_{cap}(v_0 \cdots v_n) \geq 0\}$$

and the average-bounded version as

$$\mathsf{AvgRecharge}(W, cap, t) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathrm{EL}_{cap}(v_0 \cdots v_i) \leq t\} \cap \mathsf{Recharge}(W, cap).$$

D.4.1 Solving Average-bounded Recharge Games

First, we show that solving average-bounded recharge games for a given threshold t and a given recharge capacity *cap* is EXPTIME-complete and that

the problem is still EXPTIME-hard, if the capacity is existentially quantified and only the threshold is given. Formally, we are interested in the following problems:

Problem D.3. *Solving Average-bounded recharge games*
Input: *Arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$, $W \colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$, *cap* $\in \mathbb{N}$, *and* $t \in \mathbb{N}$.
Question: *Does Player* 0 *win* $(\mathsf{A}, \mathsf{AvgRecharge}(W, cap, t))$*?*

Problem D.4. *Solving Average-bounded recharge games with existentially quantified capacity*
Input: *Arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$, $W \colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$, *and* $t \in \mathbb{N}$.
Question: *Exists cap* $\in \mathbb{N}$ *s.t. Player* 0 *wins* $(\mathsf{A}, \mathsf{AvgRecharge}(W, cap, t))$*?*

First, we consider Problem D.3.

Theorem D.6. *Solving average-bounded recharge games is EXPTIME-complete.*

We begin the proof by presenting an exponential time algorithm for solving average-bounded recharge games by reducing them to mean-payoff games, similarly to the reduction from lower- and upper-bounded energy games to mean-payoff games [21]. The mean-payoff objective is given by

$$\mathsf{MeanPayoff}(W, t) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \limsup_{n \to \infty} \frac{1}{n} \mathrm{EL}(v_0 \cdots v_{n-1}) \leq t\}.$$

Lemma D.7. *Average-bounded recharge games can be solved in exponential time.*

Proof. Fix an arena $\mathsf{A} = (V, V_0, V_1, E, v_I)$, $W \colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$, *cap* $\in \mathbb{N}$, and $t \in \mathbb{N}$. We construct a memory structure $\mathcal{M} = (M, m_I, \mathrm{Upd})$ to reduce the average-bounded recharge game to a mean-payoff game. To this end, let $M = \{0, \ldots, cap\} \cup \{\bot\}$, $m_I = cap$, $\mathrm{Upd}(\bot, (v, v')) = \bot$, and

$$\mathrm{Upd}(c, (v, v')) = \begin{cases} cap & \text{if } W(v, v') = \mathrm{R}, \\ c + W(v, v') & \text{if } c + W(v, v') \geq 0, \\ \bot & \text{if } c + W(v, v') < 0. \end{cases}$$

Intuitively, the memory structure keeps track of the energy level as long as it is non-negative. If it is negative, then a sink state is reached. Finally, we define a new weight function W' by $W'((v, c), (v', m)) = c$ for every $c \in M \setminus \{\bot\}$ and $m \in M$ and $W'((v, \bot), (v', \bot)) = t + 1$.

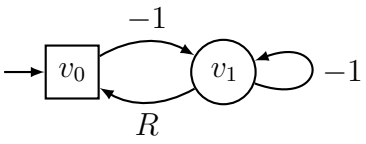


Figure D.2: The arena for the lower bound on memory requirements in average-bounded recharge games.

Remark D.8. *Let $\pi = v_0 v_1 v_2 \cdots$ and $\pi' = (v_0, m_0)(v_1, m_1)(v_2, m_2) \cdots$ be such that π is a play in A and π' is the corresponding extended play in $\mathsf{A} \times \mathcal{M}$.*

1. *If there is no $s \leq n$ such that $\mathrm{EL}_{cap}(v_0 \cdots v_s) < 0$, then $m_n = \mathrm{EL}_{cap}(v_0 \cdots v_n)$.*
2. *If there is an $s \leq n$ such that $\mathrm{EL}_{cap}(v_0 \cdots v_s) < 0$, then $m_n = \bot$.*
3. *If there is no s such that $\mathrm{EL}_{cap}(v_0 \cdots v_s) < 0$, then*

$$\limsup_{n\to\infty} \frac{1}{n} \mathrm{EL}((v_0, m_0) \cdots (v_{n-1}, m_{n-1})) = \limsup_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \mathrm{EL}_{cap}(v_0 \cdots v_i).$$

4. *If there is an s such that $\mathrm{EL}_{cap}(v_0 \cdots v_s) < 0$, then*

$$\limsup_{n\to\infty} \frac{1}{n} \mathrm{EL}((v_0, m_0) \cdots (v_{n-1}, m_{n-1}) = t + 1.$$

5. *$\pi \in \mathsf{AvgRecharge}(W, cap, t)$ if, and only if, $\pi' \in \mathsf{MeanPayoff}(W', t)$.*

Thus, we have $(\mathsf{A}, \mathsf{AvgRecharge}(W, cap, t)) \leq_{\mathcal{M}} (\mathsf{A} \times \mathcal{M}, \mathsf{MeanPayoff}(W', t))$. Hence, positional determinacy of mean-payoff games [54], Lemma D.1, and mean-payoff games being solvable in pseudo-polynomial time [126] yield the exponential time algorithm. □

An application of Lemma D.1 additionally yields an upper bound on the necessary memory states to implement a winning strategy.

Corollary D.9. *If Player 0 wins an average-bounded recharge game with capacity cap, then she also wins it with a finite-state strategy of size $cap + 2$.*

Conversely, it is straightforward to show that this bound is tight: consider the average-bounded recharge game depicted in Figure D.2 with some fixed even capacity *cap* and threshold $t = \frac{cap}{2}$. With *cap* memory states, Player 0 can implement a strategy whose unique consistent play has the form $(v_0 v_1^{cap})^\omega$ which has the energy levels $(cap, cap - 1, \ldots, 1, 0)^\omega$, which results in a long-run average of t. However, with $n < cap$ memory states, the best Player 0 is able

to do is to implement a strategy whose unique consistent play has the form $(v_0 v_1^n)^\omega$ which has the energy levels $(cap, cap-1, \ldots, cap-n)^\omega$, which results in a long-run average of $(cap-n)+\frac{n}{2} = cap - \frac{n}{2} > cap - \frac{cap}{2} = t$. Every other play that is implementable with n memory states has an even higher average. Thus, Player 0 needs *cap* memory states to meet the bound on the average.

Next, we give an EXPTIME lower bound by a reduction from countdown games. The arena $\mathsf{A} = (V, V_0, V_1, E, v_I)$ and the weight function W of such a game are subject to some restrictions:

1. The initial vertex is in V_0 and there is a designated sink vertex $v_\perp \in V_1$ with a self loop,
2. every vertex in V_0 has an edge to $v_\perp$ and all other edges are in $V_0 \times (V_1 \setminus \{v_\perp\}) \cup (V_1 \setminus \{v_\perp\}) \times V_0$,
3. all edges in $V_0 \times (V_1 \setminus \{v_\perp\})$ have negative weight and there are no two outgoing transitions from a vertex in V_0 with the same weight, and
4. all other edges have weight zero.

The objective is given as

$$\mathsf{Countdown}(W, c) = \{v_0 v_1 v_2 \cdots \in V^\omega \mid \exists n.\, v_n = v_\perp \text{ and } c + EL(v_0 \cdots v_n) = 0\}.$$

Intuitively, Player 0 picks negative weights that are subtracted from the initial energy c and Player 1 picks the next vertex to continue at (vertices of the countdown game are in V_0, V_1 only contains auxiliary vertices). Player 0 wins if the energy level is exactly zero at some point, at which she has to move to the sink vertex. Otherwise, Player 1 wins. Solving countdown games is EXPTIME-complete [80]. Our reduction is a straightforward adaption of the reduction from countdown to average-energy games [21].

Lemma D.10. *Solving average-bounded recharge games is EXPTIME-hard.*

Proof. Fix $\mathsf{A} = (V, V_0, V_1, E, v_I)$ and W satisfying the requirements of a countdown game and some initial energy c. We add a fresh vertex v_I' to V_1, add an edge from v_I' to v_I and label it with the recharge action R to obtain the arena A' and the weight function W'. As every play that does not reach the sink vertex traverses infinitely many edges with negative weight, we have $\pi \in \mathsf{Countdown}(W, c)$ if, and only if, $v_I' \cdot \pi \in \mathsf{AvgRecharge}(W', c, 0)$. Thus, Player 0 wins $(\mathsf{A}', \mathsf{AvgRecharge}(W', c, 0))$ if, and only if, she wins $(\mathsf{A}, \mathsf{Countdown}(W, c))$. Hence, solving average-bounded recharge games is EXPTIME-hard. □

Note that the hardness depends on the requirement to bound the average. Recharge games without average-bound are solvable in pseudo-polynomial time, as such a game can be expressed as a one-dimensional consumption game [24]. Determining the minimal cover (the analogue of our capacity in consumption games, see [24] for a formal definition) for the initial vertex and comparing it to the given capacity yields the desired result, as the minimal cover in a one-dimensional consumption game can be computed in pseudo-polynomial-time [24]. Whether recharge games can be solved in polynomial time is open. In the next subsection, we present a variant that is solvable in polynomial time.

Also, the previous hardness proof can be adapted to recharge games with a given threshold and existentially quantified capacity (Problem D.4). To this end, we add the initial gadget presented in Figure D.3 to a countdown game $\mathcal{G}$. In order to win this game, Player 0 has to reach the Player 1 vertex with energy level c. If the energy level is larger then Player 1 can take the edge with weight $-c$ and reach the sink with a positive energy level. Hence, the average accumulated energy will be non-zero, too. Conversely, if the energy level is smaller than c, then taking the same edge yields a negative energy level. Hence, in both cases the objective $\mathsf{AvgRecharge}(W, cap, 0)$ is violated, independently of the value of c. However, if Player 0 reaches the Player 1 vertex with energy level c, then she wins from there, if and only if, she has a winning strategy for the countdown game $\mathcal{G}$ with initial value c. Thus, she wins the recharge game with objective $\mathsf{AvgRecharge}(W, cap, 0)$ for some cap if, and only if, she wins the countdown game $\mathcal{G}$ with objective $\mathsf{Countdown}(W, c)$.

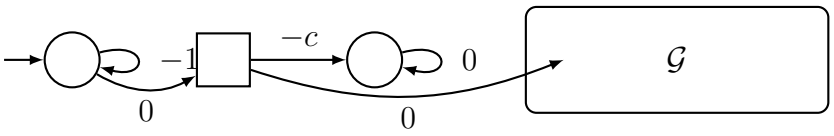


Figure D.3: The gadget for showing Problem D.4 EXPTIME-hard.

Theorem D.11. *Solving average-bounded recharge games with existentially quantified capacity and a given threshold is EXPTIME-hard.*

However, it is an open problem whether these games can be solved in exponential time. The reduction to mean-payoff games presented above depends on the capacity being part of the input. This is related to the absence of good upper bounds on the necessary capacity to achieve a given threshold.

D.4.2 Finding a Sufficient Capacity in Recharge Games

To tackle the high complexity of solving average-bounded recharge games, we consider the problem where the recharge capacity *cap* and the threshold t are existentially quantified. As the energy level is always bounded from above by *cap*, which implies that the average accumulated energy is also bounded by *cap*, it suffices to consider the objective $\mathsf{Recharge}(W, \mathit{cap})$, analogous results hold for the objective $\mathsf{AvgRecharge}(W, \mathit{cap}, t)$. We show that the following problem can be solved in polynomial time.

Problem D.5. *Existence of a sufficient recharge level in recharge games*
***Input**: Arena* $\mathsf{A} = (V, V_0, V_1, E, v_I)$ *and* $W \colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$
***Question**: Exists a capacity cap s.t. Player* 0 *wins* $(\mathsf{A}, \mathsf{Recharge}(W, \mathit{cap}))$*?*

One attempt to prove this result is to again encode the game as a one-dimensional consumption game as described above. However, this only yields a pseudo-polynomial time algorithm. In the following, we present a truly polynomial time algorithm by a reduction to three-color parity games. Given a coloring $\Omega \colon V \to \mathbb{N}$, $\mathsf{Parity}(\Omega)$ denotes the (max)-parity objective, which contains all plays $v_0 v_1 v_2 \cdots \in V^\omega$ such that the maximal color appearing infinitely often in $\Omega(v_0)\Omega(v_1)\Omega(v_2)\cdots$ is even.

Theorem D.12. *The existence of a sufficient recharge level in a recharge game can be determined in polynomial time.*

Proof. Fix an arena $\mathsf{A} = (V, V_0, V_1, E, v_I)$ and $W \colon E \to -\mathbb{N} \cup \{\mathrm{R}\}$. We construct a three-color parity game with the following property: Player 0 wins the parity game if, and only if, there is a *cap* such that Player 0 wins $(\mathsf{A}, \mathsf{Recharge}(W, \mathit{cap}))$. We assume w.l.o.g. that every vertex of A either only has incoming edges labeled with R, only has incoming edges labeled with 0, or only has incoming edges labeled with a negative weight. This can always be achieved by tripling the set of vertices, one copy for each type of incoming edge. The new initial vertex is some fixed copy of the original initial vertex. This transformation does not change the winner and only results in a linear increase in the number of states.

Now, we can speak of recharge-vertices, zero-vertices, and of decrement-vertices and define the coloring Ω such that it assigns color 2 to the recharge-vertices, color 1 to the decrement-vertices, and color 0 to the zero-vertices. We claim that Player 0 has a winning strategy for the induced parity game if, and only if, there is a *cap* such that Player 0 wins $(\mathsf{A}, \mathsf{Recharge}(W, \mathit{cap}))$.

First, assume Player 0 has a winning strategy for the parity game, which we

can assume w.l.o.g. to be positional [56, 100]. Let W be the largest absolute weight in the image of W and define $cap = (|V| - 1) \cdot W$. We claim that σ is a winning strategy for Player 0 in $(\mathsf{A}, \mathsf{Recharge}(W, cap))$. Assume it is not: then, there is a play prefix $v_0 \cdots v_n$ that is consistent with σ such that $\mathrm{EL}_{cap}(v_0 \cdots v_n) < 0$. Let $v_i \cdots v_n$ be the suffix since the last recharge edge was traversed, i.e., $-\mathrm{EL}(v_i \cdots v_n) > cap$. By the choice of cap, there are positions j and j' satisfying $i < j < j' \leq n$ such that $v_j = v_{j'}$ and $\mathrm{EL}(v_j \cdots v_{j'}) < 0$, i.e., there is a cycle with negative cost and without recharge edge. As σ is positional, the play $v_0 \cdots v_{j-1}(v_j \cdots v_{j'-1})^\omega$ obtained by reaching and then repeating this cycle is consistent with σ as well. However, in the parity game, this cycle visits no recharge-vertex, but at least one decrement-vertex. Hence, it is losing for Player 0, which contradicts σ being a winning strategy. Hence, σ is indeed also a winning strategy for $(\mathsf{A}, \mathsf{Recharge}(W, cap))$.

Now, assume there is some cap and a strategy σ that is winning for Player 0 in $(\mathsf{A}, \mathsf{Recharge}(W, cap))$. We claim that this strategy is also winning for her in the parity game. Assume, it is not, i.e., there is a play that is consistent with σ, but losing for Player 0 in the parity game. By our choice of colors, this implies that this play visits only finitely many recharge-vertices, but infinitely many decrement-vertices. Thus, it has a prefix whose recharge energy level is negative. But this contradicts the fact that σ is a winning strategy for the recharge game.

To conclude, it remains to remark that three-color parity games can be solved in polynomial time. □

By applying both directions of the equivalence, we obtain the following corollary.

Corollary D.13. *If there is a cap such that Player* 0 *wins* $(\mathsf{A}, \mathsf{Recharge}(W, cap))$*, then she also wins* $(\mathsf{A}, \mathsf{Recharge}(3 \cdot (n-1) \cdot W, W))$*, where* n *is the number of vertices of* A *and* W *is the largest absolute weight in the domain of* W*. Player* 0 *wins the latter game with a finite-state strategy of size three.*

Note that this can be improved slightly by a finer analysis: the factor $(n - 1)$ can be replaced by the number of decrement-vertices. Conversely, it is straightforward to construct examples that prove these bounds to be tight, e.g., a cycle of n edges, one being a recharge edge and all others having weight $-W$.

D.5 Tradeoffs in Recharge Games

In this section, we illustrate two different tradeoff scenarios between different quality measures for winning strategies that occur in average-bounded recharge games, i.e., tradeoffs between capacity and long-run average and between memory size and long-run average. Note that increasing the recharge capacity in such a game has a (possibly negative) influence on the long-run average, as every recharge returns the energy level to the capacity. All games we consider here are solitaire games for Player 0, i.e., every vertex belongs to Player 0. Thus, a strategy can be identified with the unique play consistent with it.

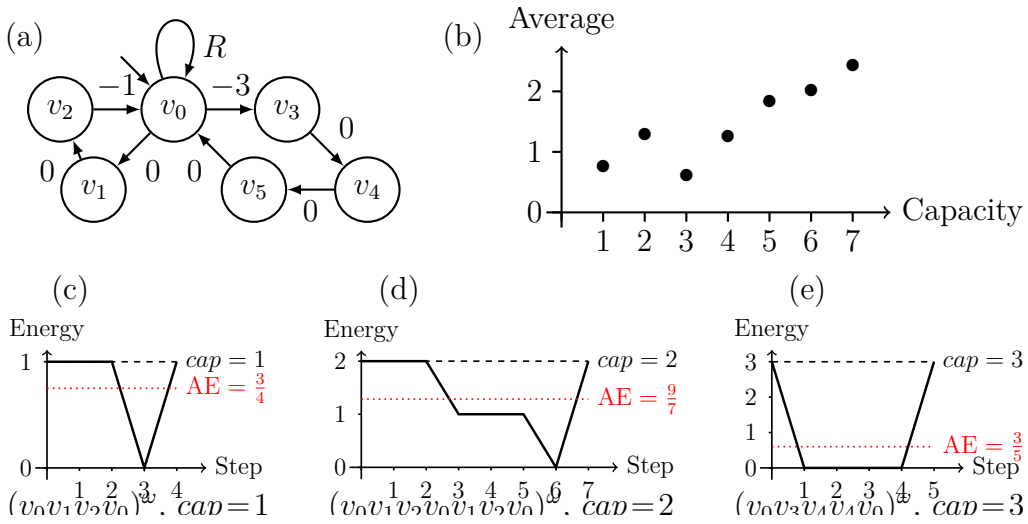


Figure D.4: (a) An average-bounded recharge game with tradeoff between capacity and long-run average. (b) A plot of the tradeoff. (c) - (e) Energy progressions of different plays in the average-bounded recharge game for different capacities.

First, we study the tradeoff between the capacity and the long-run average energy level. Consider the game in Figure D.4(a): Player 0 wins the game for $cap = 1$ and $t = 1$ by realizing the long-run average $\frac{3}{4}$ with the play $(v_0v_1v_2v_0)^\omega$ (Figure D.4(c)). But, by increasing the capacity to $cap = 2$, it is no longer possible for her to win for $t = 1$, as the best long-run average she can realize is $\frac{9}{7}$ by playing $(v_0v_1v_2v_0v_1v_2v_0)^\omega$ (Figure D.4(d)). However, for $cap = 3$, she can again win for $t = 1$, and it is possible to realize the long-run average $\frac{3}{5}$ by playing $(v_0v_3v_4v_5v_0)^\omega$ (Figure D.4(e)). Again, with $cap = 4$ Player 0 loses for $t = 1$.

This example shows that higher capacity can be traded for a lower long-run average and that the tradeoff is non-monotonic. Figure D.4(b) shows a plot of the tradeoff for capacities ranging from 1 to 7.

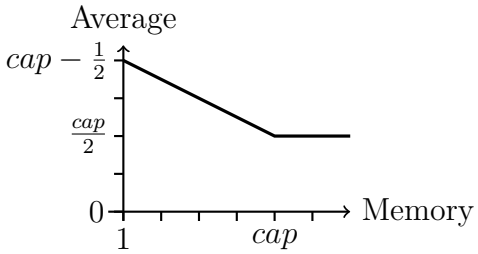


Figure D.5: A plot of the tradeoff between memory size and long-run average in the game in Figure D.2.

Another tradeoff scenario is between the number of memory states required to implement a strategy and the long-run average energy level it realizes. Consider the recharge game from Figure D.2: as discussed below Corollary D.9, Player 0 can win for the threshold $t = \frac{cap}{2}$ with cap memory states. However, with $n < cap$ memory states, she can only guarantee the long-run average $(cap - n) + \frac{n}{2}$. In particular, the best long-run average that is realizable by a positional strategy (which requires one memory state to implement) is $cap - \frac{1}{2}$ (see Fig. D.5).

D.6 Conclusion

We continued the study of average-energy games by considering problems where the bound on the average is existentially quantified instead of given as part of the input. We showed that solving this problem is equivalent to determining whether the maximal energy level can be uniformly bounded by a strategy. The latter problem is known to be decidable in doubly-exponential time, which therefore also holds for our original problem. Then, we considered a different type of energy evolution where energy is only consumed or reset to some fixed capacity. Solving the average-bounded variants of these games is shown to be complete for exponential time. Due to this high complexity, we again considered a variant where the bounds are existentially quantified. This problem turns out to be solvable in polynomial time. Finally, we studied tradeoffs between the different bounds and the memory requirements of winning strategies: increasing the upper bound on the maximal energy level is shown to allow to improve the average energy level and memory can be traded for smaller upper bounds and vice versa.

For future work, it would be interesting to extend our results to a multi-dimensional setting. Also, the exact complexity of determining the existence of

an upper bound in average-energy games is open. Finally, the decidability of average-energy games with a given threshold, but without an upper bound on the energy level is open [21]. In current work, we study whether our approach presented in Section D.3 can be adapted to solve these problems, e.g., by not picking representatives by minimizing peak height but some other measure. These questions are also related to the complexity of recharge games with a given threshold where the capacity is existentially quantified. Finally, we are studying upper bounds on the tradeoffs presented in Section D.5.

Acknowledgements. The work presented here was carried out while the third author visited the Distributed and Embedded Systems Unit at Aalborg University and while the second author visited the Reactive Systems Group at Saarland University. We thank these institutions for their hospitality, and for the support by the DFG project TriCS (ZI 1516/1-1), the ERC Advanced Grant LASSO and the EU FET projects SENSATION and CASSTING.

References

[1] Rajeev Alur, Kousha Etessami, Salvatore La Torre, and Doron Peled. Parametric temporal logic for model measuring. *ACM Trans. Comput. Log.*, 2(3):388–407, 2001.

[2] B. Aminof and S. Rubin. First cycle games. In *Proc. of SR*, EPTCS 146, pages 83–90, 2014.

[3] D.S. Ananichev and M.V. Volkov. Synchronizing monotonic automata. *Theoretical Computer Science*, 327(3):225 – 239, 2004.

[4] Esther M. Arkin, Christos H. Papadimitriou, and Mihalis Yannakakis. Modularity of cycles and paths in graphs. *J. ACM*, 38(2):255–274, 1991.

[5] David Arney, Miroslav Pajic, Julian M Goldman, Insup Lee, Rahul Mangharam, and Oleg Sokolsky. Toward patient safety in closed-loop medical device systems. In *Proceedings of the 1st ACM/IEEE International Conference on Cyber-Physical Systems*, pages 139–148. ACM, 2010.

[6] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[7] Paolo Baldan and Daniele Gorla, editors. *CONCUR 2014 - Concurrency Theory - 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings*, volume 8704 of *Lecture Notes in Computer Science*. Springer, 2014.

[8] Jørgen Bang-Jensen and Gregory Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2nd edition, 2008.

[9] Yaakov Benenson, Rivka Adar, Tamar Paz-Elizur, Zvi Livneh, and Ehud Shapiro. Dna molecule provides a computing machine with both data and fuel. *Proceedings of the National Academy of Sciences of the USA*, 100(5):2191–2196, March 2003.

[10] Mikhail V Berlinkov. On two algorithmic problems about synchronizing automata. In *International Conference on Developments in Language Theory*, pages 61–67. Springer, 2014.

[11] H. Björklund, S. Sandberg, and S. Vorobyov. Memoryless determinacy of parity and mean payoff games: A simple proof. *Theoretical Computer Science*, 310(1-3):365–378, 2004.

[12] R. Bloem, K. Chatterjee, T.A. Henzinger, and B. Jobstmann. Better

quality in synthesis through quantitative objectives. In *Proc. of CAV*, LNCS 5643, pages 140–156. Springer, 2009.

[13] Thomas Boel. Årsag til fejl på Aalborg-satellit: Solcellerne vendte væk fra solen. Ingeniøren (Weekly national news magazine about engineering), 8. March 2013. https://ing.dk/artikel/aarsag-til-fejl-paa-aalborg-satellit-solcellerne-vendte-vaek-fra-solen-156828.

[14] A. Bohy, V. Bruyère, E. Filiot, and J.-F. Raskin. Synthesis from LTL specifications with mean-payoff objectives. In *Proc. of TACAS*, LNCS 7795, pages 169–184. Springer, 2013.

[15] Udi Boker, Thomas A. Henzinger, and Arjun Radhakrishna. Battery transition systems. *SIGPLAN Not.*, 49(1):595–606, January 2014.

[16] Emile Borel and Jean Ville. *Applications aux jeux de hasard*. Gauthier-Vilars, 1938.

[17] Endre Boros, Khaled M. Elbassioni, Vladimir Gurvich, and Kazuhisa Makino. Markov decision processes and stochastic games with total effective payoff. In Ernst W. Mayr and Nicolas Ollinger, editors, *STACS 2015*, volume 30 of *LIPIcs*, pages 103–115. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[18] P. Bouyer, U. Fahrenberg, K.G. Larsen, N. Markey, and J. Srba. Infinite runs in weighted timed automata with energy constraints. In *Proc. of FORMATS*, LNCS 5215, pages 33–47. Springer, 2008.

[19] P. Bouyer, N. Markey, M. Randour, K.G. Larsen, and S. Laursen. Average-energy games. In *Proc. of GandALF*, EPTCS 193, pages 1–15, 2015.

[20] Patricia Bouyer, Uli Fahrenberg, Kim G. Larsen, Nicolas Markey, and Jiří Srba. Infinite runs in weighted timed automata with energy constraints. In Franck Cassez and Claude Jard, editors, *Formats 2008*, volume 5215 of *LNCS*, pages 33–47. Springer, 2008.

[21] Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim G. Larsen, and Simon Laursen. Average-energy games. In Javier Esparza and Enrico Tronci, editors, *GandALF 2015*, volume 193 of *EPTCS*, pages 1–15. Open Publishing Association, 2015.

[22] Patricia Bouyer, Nicolas Markey, Mickael Randour, Kim G. Larsen, and Simon Laursen. Average-energy games. *Acta Informatica*, pages 1–37, 2016.

[23] T. Brázdil, D. Klaška, A. Kučera, and P. Novotný. Minimizing running

costs in consumption systems. In *Proc. of CAV*, LNCS 8559, pages 457–472. Springer, 2014.

[24] Tomás Brázdil, Krishnendu Chatterjee, Antonín Kucera, and Petr Novotný. Efficient controller synthesis for consumption games with multiple resource types. In P. Madhusudan and Sanjit A. Seshia, editors, *CAV 2012*, volume 7358 of *LNCS*, pages 23–38. Springer, 2012.

[25] Thomas Brihaye, Gilles Geeraerts, Axel Haddad, Benjamin Monmege, Guillermo A. Pérez, and Gabriel Renault. Quantitative games under failures. In Prahladh Harsha and G. Ramalingam, editors, *35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India*, volume 45 of *LIPIcs*, pages 293–306. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[26] L. Brim, J. Chaloupka, L. Doyen, R. Gentilini, and J.-F. Raskin. Faster algorithms for mean-payoff games. *Formal Methods in System Design*, 38(2):97–118, 2011.

[27] Manfred Broy, Bengt Jonsson, Joost-Pieter Katoen, Martin Leucker, and Alexander Pretschner, editors. *Model-Based Testing of Reactive Systems, Advanced Lectures*, volume 3472 of *Lecture Notes in Computer Science*. Springer, 2005.

[28] Franck Cassez, Alexandre David, KimG. Larsen, Didier Lime, and Jean-François Raskin. Timed control with observation based and stuttering invariant strategies. In *Automated Technology for Verification and Analysis*, volume 4762 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2007.

[29] Franck Cassez, Jan J. Jensen, Kim G. Larsen, Jean-François Raskin, and Pierre-Alain Reynier. Automatic synthesis of robust and optimal controllers – an industrial case study. In Rupak Majumdar and Paulo Tabuada, editors, *HSCC 2009*, volume 5469 of *LNCS*, pages 90–104. Springer, 2009.

[30] Ján Černý. Poznámka k. homogénnym experimentom s konecnými automatmi. *Mat. fyz. čas SAV*, 14:208–215, 1964.

[31] Arindam Chakrabarti, Luca de Alfaro, Thomas A. Henzinger, and Mariëlle Stoelinga. Resource interfaces. In Rajeev Alur and Insup Lee, editors, *EMSOFT 2003*, volume 2855 of *LNCS*, pages 117–133. Springer, 2003.

[32] Satish Chandra, Patrice Godefroid, and Christopher Palm. Software model checking in practice: An industrial case study. In *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, pages 431–441, New York, NY, USA, 2002. ACM.

[33] K. Chatterjee, L. Doyen, M. Randour, and J.-F. Raskin. Looking at mean-payoff and total-payoff through windows. *Information and Computation*, 242:25–52, 2015.

[34] K. Chatterjee, T.A. Henzinger, and M. Jurdziński. Mean-payoff parity games. In *Proc. of LICS*, pages 178–187. IEEE Comp. Soc. Press, 2005.

[35] K. Chatterjee and V.S. Prabhu. Quantitative timed simulation functions and refinement metrics for real-time systems. In *Proc. of HSCC*, pages 273–282. ACM, 2013.

[36] K. Chatterjee, M. Randour, and J.-F. Raskin. Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3-4):129–163, 2014.

[37] K. Chatterjee and Y. Velner. Mean-payoff pushdown games. In *Proc. of LICS*, pages 195–204. IEEE Computer Society, 2012.

[38] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. In Samson Abramsky, Cyril Gavoille, Claude Kirchner, Friedhelm Meyer auf der Heide, and Paul G. Spirakis, editors, *Proceedings of the 37th International Colloquium on Automata, Languages and Programming (ICALP'10) – Part II*, volume 6199 of *Lecture Notes in Computer Science*, pages 599–610. Springer-Verlag, July 2010.

[39] Krishnendu Chatterjee and Laurent Doyen. Energy parity games. *Theor. Comput. Sci.*, 458:49–60, 2012.

[40] Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, and Jean françois Raskin. Algorithms for omega-regular games with imperfect information. In *CSL'06*, volume 4207 of *LNCS*, pages 287–302. Springer, 2006.

[41] Krishnendu Chatterjee and Thomas A Henzinger. Semiperfect-information games. In *FSTTCS'05*, pages 1–18. Springer, 2005.

[42] Krishnendu Chatterjee, Thomas A. Henzinger, and Marcin Jurdziński. Mean-payoff parity games. In *LICS 2005*, pages 178–187. IEEE Computer Society, 2005.

[43] Krishnendu Chatterjee, Mickael Randour, and Jean-François Raskin.

Strategy synthesis for multi-dimensional quantitative objectives. *Acta Informatica*, 51(3):129–163, 2013.

[44] Alonzo Church. Applications of recursive arithmetic to the problem of circuit synthesis. *Summaries of the Summer Institute of Symbolic Logic*, 1:3–50, 1957.

[45] Edmund M. Clarke and E. Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In Dexter Kozen, editor, *Logics of Programs*, volume 131 of *Lecture Notes in Computer Science*, pages 52–71. Springer Berlin Heidelberg, 1982.

[46] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 8(2):244–263, 1986.

[47] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.

[48] Antoine-Augustin Cournot. Recherches sur les principes mathématiques de la théorie des richesses [researches into the mathematical principles of the theory of wealth]. *L. Hachette*, 1838. Original paper in French, English version by Macmillan, New York, 1897. (Reprinted Augustus M. Kelley, New York, 1971).

[49] Laurent Doyen, Line Juhl, Kim Guldstrand Larsen, Nicolas Markey, and Mahsa Shirmohammadi. Synchronizing words for weighted and timed automata. In Venkatesh Raman and S. P. Suresh, editors, *34th International Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2014, December 15-17, 2014, New Delhi, India*, volume 29 of *LIPIcs*, pages 121–132. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2014.

[50] Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Infinite synchronizing words for probabilistic automata. In *MFCS'11*, volume 6907 of *LNCS*, pages 278–289. Springer, 2011.

[51] Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Synchronizing objectives for markov decision processes. In Johannes Reich and Bernd Finkbeiner, editors, *iWIGP*, volume 50 of *EPTCS*, pages 61–75, 2011.

[52] Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Limit synchronization in markov decision processes. In Anca Muscholl, editor,

Foundations of Software Science and Computation Structures - 17th International Conference, FOSSACS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014, Proceedings, volume 8412 of *Lecture Notes in Computer Science*, pages 58–72. Springer, 2014.

[53] Laurent Doyen, Thierry Massart, and Mahsa Shirmohammadi. Robust synchronization in markov decision processes. In Baldan and Gorla [7], pages 234–248.

[54] A. Ehrenfeucht and J. Mycielski. Positional strategies for mean payoff games. *International Journal of Game Theory*, 8:109–113, 1979.

[55] Daniel Ejsing-Dunn and Lisa Fontani. Infinite runs in recharge automata. Master's thesis, Computer Science Department, Aalborg University, Denmark, 2013. Available at http://www.cassting-project.eu/wp-content/uploads/master13-EF.pdf.

[56] E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *FOCS 1991*, pages 368–377. IEEE, 1991.

[57] E.Allen Emerson and EdmundM. Clarke. Characterizing correctness properties of parallel programs using fixpoints. In Jaco de Bakker and Jan van Leeuwen, editors, *Automata, Languages and Programming*, volume 85 of *Lecture Notes in Computer Science*, pages 169–181. Springer Berlin Heidelberg, 1980.

[58] D. Eppstein. Reset sequences for monotonic automata. *SIAM Journal on Computing*, 19(3):500–510, 1990.

[59] Uli Fahrenberg, Line Juhl, Kim G. Larsen, and Jirí Srba. Energy games in multiweighted automata. In Antonio Cerone and Pekka Pihlajasaari, editors, *ICTAC*, volume 6916 of *Lecture Notes in Computer Science*, pages 95–115. Springer, 2011.

[60] Peter Faymonville and Martin Zimmermann. Parametric linear dynamic logic. In Adriano Peron and Carla Piazza, editors, *GandALF 2014*, volume 161 of *EPTCS*, pages 60–73, 2014.

[61] J. Fearnley and M. Jurdziński. Reachability in two-clock timed automata is PSPACE-complete. In *Proc. of ICALP*, LNCS 7966, pages 212–223. Springer, 2013.

[62] J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer, 1997.

[63] Limor Fix. Fifteen years of formal property verification in intel. In *25 Years of Model Checking*, pages 139–144. Springer, 2008.

[64] Fedor Fominykh and Mikhail Volkov. P(l)aying for synchronization. In *Implementation and Application of Automata*, volume 7381 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 2012.

[65] M.R. Garey and D.S. Johnson. *Computers and intractability: a guide to the Theory of NP-Completeness*. Freeman New York, 1979.

[66] T. Gawlitza and H. Seidl. Games through nested fixpoints. In *Proc. of CAV*, LNCS 5643, pages 291–305. Springer, 2009.

[67] Arthur Gill. State-identification experiments in finite automata. *Information and Control*, 4(2–3):132 – 154, 1961.

[68] H. Gimbert and W. Zielonka. When can you play positionnaly? In *Proc. of MFCS*, LNCS 3153, pages 686–697. Springer, 2004.

[69] H. Gimbert and W. Zielonka. Games where you can play optimally without any memory. In *Proc. of CONCUR*, LNCS 3653, pages 428–442. Springer, 2005.

[70] Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

[71] Ramesh Hariharan, Telikepalli Kavitha, and Kurt Mehlhorn. Faster algorithms for minimum cycle basis in directed graphs. *SIAM J. Comput.*, 38(4):1430–1447, 2008.

[72] K. Havelund, A. Skou, K. G. Larsen, and K. Lund. Formal modeling and analysis of an audio/video protocol: an industrial case study using uppaal. In *Real-Time Systems Symposium, 1997. Proceedings., The 18th IEEE*, pages 2–13, Dec 1997.

[73] Thomas A Henzinger and Joseph Sifakis. The embedded systems design challenge. In *FM 2006: Formal Methods*, pages 1–15. Springer, 2006.

[74] Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

[75] Charles Antony Richard Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[76] D.R. Hofstadter. *Godel, Escher, Bach: An Eternal Golden Braid*. Basic Books. Basic Books, 1999.

[77] Szabolcs Iván. Synchronizing weighted automata. In Zoltán Ésik and Zoltán Fülöp, editors, *Proceedings 14th International Conference on Automata and Formal Languages, AFL 2014, Szeged, Hungary, May 27-29, 2014.*, volume 151 of *EPTCS*, pages 301–313, 2014.

[78] Line Juhl, Kim G. Larsen, and Jean-François Raskin. Optimal bounds for multiweighted and parametrised energy games. In Zhiming Liu, Jim Woodcock, and Huibiao Zhu, editors, *Theories of Programming and Formal Methods - Essays Dedicated to Jifeng He on the Occasion of His 70th Birthday*, volume 8051 of *LNCS*, pages 244–255. Springer, 2013.

[79] M. Jurdziński. Deciding the winner in parity games is in UP∩co-UP. *Information Processing Letters*, 68(3):119–124, 1998.

[80] Marcin Jurdziński, Jeremy Sproston, and François Laroussinie. Model checking probabilistic timed automata with one or two clocks. *LMCS*, 4(3), 2008.

[81] Cem Kaner, Jack Falk, and Hung Quoc Nguyen. *Testing Computer Software Second Edition*. Dreamtech Press, 2000.

[82] R.M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23(3), 1978.

[83] Robert M. Keller. Formal verification of parallel programs. *Commun. ACM*, 19(7):371–384, July 1976.

[84] Donald Knuth. Strong components. Technical Report 004639, Comput. Sci. Dept., Stanford University, Stanford, Calif., 1973.

[85] Dénes König. Über eine schlussweise aus dem endlichen ins unendliche. *Acta Litt. ac. sci. Szeged*, 3:121–130, 1927.

[86] E. Kopczynski. Half-positional determinacy of infinite games. In *Proc. of ICALP*, LNCS 4052, pages 336–347. Springer, 2006.

[87] S. Rao Kosaraju and Gregory F. Sullivan. Detecting cycles in dynamic graphs in polynomial time (preliminary version). In Janos Simon, editor, *Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2-4, 1988, Chicago, Illinois, USA*, pages 398–406. ACM, 1988.

[88] Dexter Kozen. Lower bounds for natural proof systems. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA, 31 October - 1 November 1977*, pages 254–266. IEEE Computer Society, 1977.

[89] Jan Kretínský, Kim Guldstrand Larsen, Simon Laursen, and Jirí Srba.

Polynomial time decidability of weighted synchronization under partial observability. In Luca Aceto and David de Frutos-Escrig, editors, *26th International Conference on Concurrency Theory, CONCUR 2015, Madrid, Spain, September 1.4, 2015*, volume 42 of *LIPIcs*, pages 142–154. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2015.

[90] Moez Krichen. State identification. In Broy et al. [27], pages 35–67.

[91] Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. From liveness to promptness. *Formal Methods in System Design*, 34(2):83–103, 2009.

[92] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for AC-like equational theories with homomorphisms. Research Report LSV-04-16, Laboratoire Spécification et Vérification, ENS Cachan, France, 2004.

[93] P. Lafourcade, D. Lugiez, and R. Treinen. Intruder deduction for *AC*-like equational theories with homomorphisms. In *Proc. of RTA*, LNCS 3467, pages 308–322. Springer, 2005.

[94] K.G. Larsen, S. Laursen, and M. Zimmermann. Limit your consumption! Finding bounds in average-energy games. *CoRR*, abs/1510.05774, 2015.

[95] Kim Guldstrand Larsen, Simon Laursen, and Jiří Srba. Synchronizing strategies under partial observability. In Baldan and Gorla [7], pages 188–202.

[96] Oded Maler, Amir Pnueli, and Joseph Sifakis. On the synthesis of discrete controllers for timed systems. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 229–242. Springer, 1995.

[97] Pavel Martyugin. Computational complexity of certain problems related to carefully synchronizing words for partial automata and directing words for nondeterministic automata. *Theory of Com. Systems*, pages 1–12, 2013.

[98] Robin Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer, 1980.

[99] Edward F. Moore. Gedanken Experiments on Sequential Machines. In *Automata Studies*, pages 129–153. Princeton U., 1956.

[100] Andrzej Mostowski. Games with forbidden positions. Technical Report 78, University of Gdańsk, 1991.

[101] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.

[102] Gethin Norman, David Parker, and Xueyi Zou. Verification and control of partially observable probabilistic real-time systems. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 240–255. Springer, 2015.

[103] Jörg Olschewski and Michael Ummels. The complexity of finding reset words in finite automata. In Petr Hlinený and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2010, 35th International Symposium, MFCS 2010, Brno, Czech Republic, August 23-27, 2010. Proceedings*, volume 6281 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2010.

[104] Susan Owicki and David Gries. Verifying properties of parallel programs: An axiomatic approach. *Communications of the ACM*, 19(5):279–285, 1976.

[105] Guillermo A. Pérez. The fixed initial credit problem for energy games with partial-observation is ackermann-complete. *CoRR*, abs/1512.04255, 2015.

[106] Jean-Eric Pin. On two combinatorial problems arising from automata theory. *North-Holland Mathematics Studies*, 75:535–548, 1983.

[107] A. Pnueli and R. Rosner. On the synthesis of a reactive module. In *Proceedings of the 16th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, POPL '89, pages 179–190, New York, NY, USA, 1989. ACM.

[108] Irith Pomeranz and Sudhakar M. Reddy. Application of homing sequences to synchronous sequential circuit testing. *IEEE Trans. Computers*, 43(5):569–580, 1994.

[109] J.P. Queille and J. Sifakis. Specification and verification of concurrent systems in cesar. In Mariangiola Dezani-Ciancaglini and Ugo Montanari, editors, *International Symposium on Programming*, volume 137 of *Lecture Notes in Computer Science*, pages 337–351. Springer Berlin Heidelberg, 1982.

[110] P. Ramadge and W. Wonham. Supervisory Control of a Class of Discrete Event Processes. *Siam J. Control and Optimization*, 25(1), 1987.

[111] M. Randour. Automated synthesis of reliable and efficient systems through game theory: A case study. In *Proceedings of the European Conference on Complex Systems 2012*, Springer Proceedings in Complexity XVII, pages 731–738. Springer, 2013.

[112] M. Randour. *Synthesis in Multi-Criteria Quantitative Games*. PhD thesis, University of Mons, Belgium, 2014.

[113] John H Reif. The complexity of two-player games of incomplete information. *Journal of computer and system sciences*, 29(2):274–301, 1984.

[114] Jussi Rintanen. Complexity of conditional planning under partial observability and infinite executions. In *ECAI*, pages 678–683, 2012.

[115] I.K. Rystsov. Polynomial complete problems in automata theory. *Information Processing Letters*, 16(3):147–151, 1983.

[116] I.K. Rystsov. Rank of a finite automaton. *Cybernetics and Systems Analysis*, 28(3):323–328, 1992.

[117] Sven Sandberg. Homing and synchronizing sequences. In Broy et al. [27], pages 5–33.

[118] M. Sipser. *Introduction to the Theory of Computation*. Course Technology, 2006.

[119] A Prasad Sistla and Edmund M Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM (JACM)*, 32(3):733–749, 1985.

[120] F. Thuijsman and O.J. Vrieze. The bad match; a total reward stochastic game. *OR Spektrum*, 9(2):93–99, 1987.

[121] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Moshe Rabinovich, and Jean-François Raskin. The complexity of multi-mean-payoff and multi-energy games. *Inf. Comput.*, 241:177–196, 2015.

[122] Mikhail V Volkov. Synchronizing automata and the černỳ conjecture. In *Language and automata theory and applications*, pages 11–27. Springer, 2008.

[123] John von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 1944.

[124] Ernst Zermelo. Über eine anwendung der mengenlehre auf die theorie des schachspiels. In *Proceedings of the fifth international congress of mathematicians*, volume 2, pages 501–504. II, Cambridge UP, Cambridge, 1913.

[125] Martin Zimmermann. Parameterized linear temporal logics meet costs:

Still not costlier than LTL. In Javier Esparza and Enrico Tronci, editors, *GandALF 2015*, volume 193 of *EPTCS*, pages 144–157. Open Publishing Association, 2015.

[126] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1-2):343–359, 1996.

ISSN (online): 2246-1248
ISBN (online): 978-87-7112-801-7

AALBORG UNIVERSITY PRESS